VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

POKROČILÁ EVOLUČNÍ FILTRACE OBRAZU

ADVANCED EVOLUTIONARY IMAGE FILTERING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. IVANA SARANOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL BIDLO, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

144305

Ústav:	Ústav počítačových systémů (UPSY)
Studentka:	**Saranová Ivana, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Vývoj aplikací
Název:	**Pokročilá evoluční filtrace obrazu**
Kategorie:	Umělá inteligence
Akademický rok:	2022/23

Zadání:

1. Seznamte se s problematikou rekonstrukce obrazu po různých formách jeho poškození.
2. Seznamte se s existujícími metodami evolučního návrhu obrazových filtrů a zpracujte studii na toto téma.
3. Zvolte, případně navrhněte vlastní, metodu pro návrh obrazových filtrů pomocí evolučních algoritmů.
4. Implementujte evoluční systém pro automatický návrh obrazových filtrů pro účely rekonstrukce obrazu po různých typech poškození.
5. Pro vybrané scénáře realizujte sady experimentů demonstrujících schopnosti navrženého přístupu.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:
Dle pokynů vedoucího projektu.

Při obhajobě semestrální části projektu je požadováno:
Splnění bodů 1 a 2 zadání, demonstrace prototypu systému k bodům 3 a 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Bidlo Michal, Ing., Ph.D.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	31.10.2022

Abstrakt

Tato práce se zaměřuje na použití celulárních automatů s přechodovou funkcí složenou z podmínkových přechodových pravidel navržených evoluční strategií pro odstranění šumů různých typů a intenzit z digitálního obrazu. Navržená metoda vylepšuje původní koncept podmínkových přechodových pravidel úpravou pravé strany pravidla a rozšiřuje ji z jedné hodnoty na výběr výpočetních funkcí. Dále byla zkoumána různá nastavení evoluční strategie, trénování na různých typech šumů, trénování na částečně poškozených obrazech, a další nastavení, což vedlo k získání kvalitních filtrů pro každý model šumu. Porovnání těchto filtrů se stávajícími metodami ukazuje velké zlepšení oproti původnímu přístupu a schopnost evolučně navrhovat filtry, které se řadí mezi ty kvalitnější mezi porovnávanými metodami.

Abstract

This work aims to use cellular automata with a transition function of conditionally matching rules designed by the evolution strategy for the removal of noises of different types and intensities from digital images. The proposed method improves the original concept of conditionally matching rules by modifying the right side of the rule, extending it from a single value to a selection of functions. Furthermore, various evolution strategy setups were explored, including usage of different noise models for evolution, training on partially damaged images, and other setups, resulting in high-quality filters for each noise model. Comparing these filters to the existing methods shows great improvement from the original approach and the ability to evolutionarily design filters that are placed among the top methods quality-wise.

Klíčová slova

obrazové filtry, celulární automat, evoluční strategie, evoluční návrh, šum typu sůl a pepř, impulse burst, náhodný šum, podmínková přechodová pravidla

Keywords

image filters, cellular automata, evolution strategy, evolutionary design, salt and pepper noise, impulse burst, random noise, conditionally matching rules

Citace

SARANOVÁ, Ivana. *Pokročilá evoluční filtrace obrazu.* Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Pokročilá evoluční filtrace obrazu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana Ing. Michala Bidla, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Ivana Saranová
14. května 2023

Poděkování

Chtěla bych poděkovat panu Ing. Michalu Bidlovi, Ph.D. za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat. Dále bych ráda poděkovala i mému kolegovi Bc. Petrovi Kubicovi za pomoc při zprovoznění prostředí pro efektivní běh a vyhodnocování experimentů, a Mgr. Tomáši Károvi za podporu a pomoc při gramatické kontrole práce.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Digitální obrazy a jejich úpravu považujeme za běžnou součást našeho života, ať už se jedná o rozpoznávání obličejů z fotografií na sociálních sítích nebo úpravu kvality pořízených snímků. Ostatně v soukromé i profesní sféře roste potřeba vysoké kvality digitálního obrazu. Naneštěstí je součástí digitálních obrazů i šum, tedy poškození obrazu, které se vyskytuje při jeho tvorbě, kompresi nebo přenosu. Ztráta nebo pozměnění původních informací může do velké míry ovlivnit zpracování obrazu. Filtrace šumu proto představuje v dnešní době jednu ze základních úloh zpracování obrazu, která předchází aplikacím pokročilejších algoritmů.

Se snahami odstranit šum se setkáme ve zdravotnictví, zejména u rentgenových snímků nebo záznamů magnetické rezonance [20]. Zpřesnění informací získaných z digitálních obrazů v této sféře může vést k lepší lékařské péči. Další oblastí, kde najde filtrace šumu praktické využití, jsou sledovací a bezpečnostní systémy. Identifikace pachatelů, detekce objektů nebo pohybu je mnohem snadnější z kvalitnějších záznamů z kamer [13]. Důležitá je i úprava satelitních snímků [19], které mají velmi široké využití napříč zemědělstvím, průmyslem a službami, a které jsou také poškozeny nejrůznějšími šumy.

Konvenční metody odstranění šumu, jako jsou například metody pracující s posuvným oknem o pevné šířce (nejčastěji 3x3 nebo 5x5) a mediánem (průměrem, majoritní hodnotou) [37], jsou dobře známé a hojně využívané. Jejich nevýhodou je zpravidla ztráta původních informací obrazu zahlazením (zprůměrováním). Alternativou je využití celulárních automatů, kterými se zabývá kapitola 2. Dvoudimenzionální celulární automaty operují s jednotlivými pixely jako buňkami a aplikují na ně přechodové funkce, čímž dochází k filtraci. Variabilita celulárních automatů umožňuje neustále zdokonalovat vlastnosti pravidel, a tudíž i zkvalitňovat odstranění šumu.

Zvolení vhodné sady pravidel pro celulární automat je však problematické. Častý problém je příliš velká specializace pravidel, takže nemohou být použita například na různé typy šumů, nebo jejich paměťová a výpočetní náročnost. Podmínková přechodová pravidla (detailně popsaná v podkapitole 2.3) přináší možnost snížit množství pravidel a zároveň si částečně zachovat složitost původních jednoduchých pravidel. Nalezení patřičných podmínkových přechodových pravidel pak může provést evoluční algoritmus s vhodným nastavením, čímž se zabývá kapitola 3.

Cílem této práce je vylepšit již existující metodu využívající evoluční strategie pro získání podmínkových přechodových pravidel, kterými je pak prováděna filtrace přes celulární automat [8]. Jedná se tedy především o experimentální práci, jejíž náplní je najít vhodné nastavení evoluce a celulárního automatu pro získání kvalitnějších filtrů. Návrh zlepšení původní metody spočívá v modifikaci podmínkových přechodových pravidel a experimentování nad různými typy poškození obrazu. Navrhovaná metoda je rozebrána v kapitole 4, přičemž

je vysvětlen hlavně princip úpravy podmínkových pravidel. Potenciál metody je i v aplikaci na různých typech a intenzitách šumu, které jsou blíže popsány v podkapitole 4.1. Nastavení a způsob vyhodnocování experimentů jsou vysvětleny v kapitole 5. V této kapitole jsou také shrnuty dosažené výsledky včetně porovnání s existujícími metodami a praktickými ukázkami.

Kapitola 2

Celulární automaty

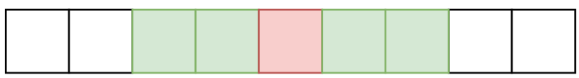
V této kapitole bude shrnut základní princip fungování celulárních automatů (definice, sousedství, přechodová pravidla) s přehledem známějších variací. Dále budou detailně rozebrána podmínková přechodová pravidla. Na konci kapitoly je pak přehled metod využívajících celulární automaty k filtrování obrazu.

2.1 Princip

Celulární automaty (CA) jsou diskrétní, deterministické matematické systémy, pro které je typická lokální interakce a paralelní evoluce [16]. Celulární automat se skládá z elementů (buněk) a přechodové funkce. Buňky jsou uspořádány do mřížky podle požadované dimenze a každá buňka nabývá právě jednoho stavu z konečné množiny možných stavů. Vstupem lokální přechodové funkce buňky je kombinace stavů buněk z lokálního sousedství a výstupem je nový stav buňky. Pro okrajové buňky mohou být definovaná speciální podmínková pravidla nebo se okraje propojují (v 1D do cyklu, ve 2D do anuloidu, atp.). Jako krok CA (iteraci) označujeme děj, kdy všechny buňky synchronně aplikují stejnou přechodovou funkci.

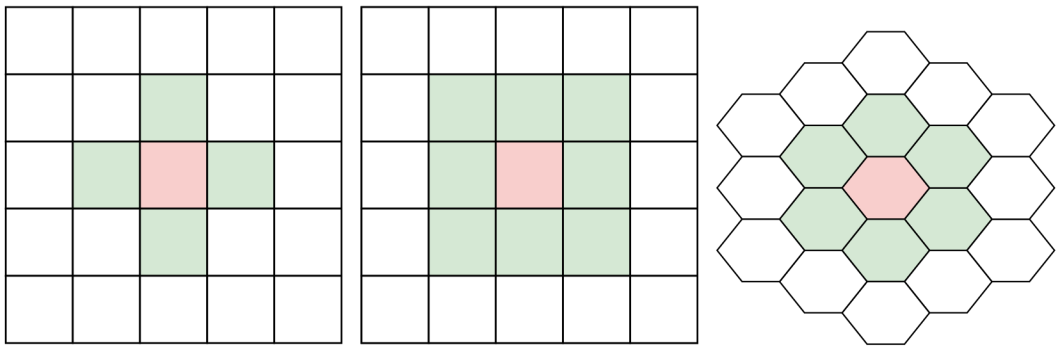
Celulární automaty mají široké spektrum využití, ale nejčastěji se používají jako modely pro nejrůznější situace z reálného světa, které se pak dále studují nebo se na nich simulují vybrané děje. Jedná se například o simulace feromagnetismu [5], chování plynů [32], modelování biologických procesů, organismů nebo ekosystémů (neurony, vzory na povrchu schránky některých druhů mlžů, vývoj a chování rakovinných buněk) [14, 31], generování pseudonáhodných čísel [17], modelování a simulace chemických reakcí [25], generování umění [1] a mnoho dalších.

Sousedství



Obrázek 2.1: Příklad 1D sousedství pro $r = 2$

Sousedství v 1D CA jsou buňky ležící do určité vzdálenosti nalevo a napravo od aktuální buňky včetně středové buňky. Na obrázku 2.1 je příklad sousedství v 1D CA s 4 sousedními buňkami. Na obrázku 2.2 jsou znázorněny nejčastější typy sousedství v 2D celulárních

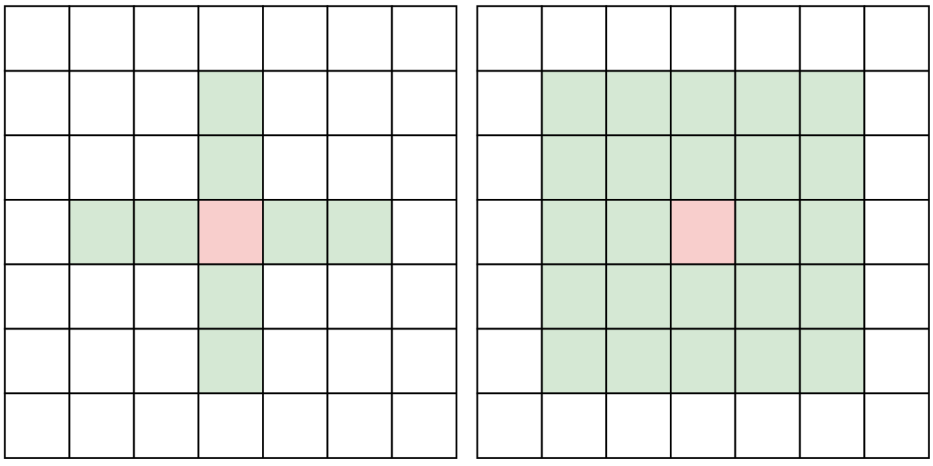


(a) (b) (c)

Obrázek 2.2: Příklady 2D sousedství pro $r = 1$, a) von Neumannovo, b) Moorovo, c) hexagonální

automatech, von Neumannovo sousedství (horní, dolní, levý a pravý soused a středová aktuální buňka označená červeně), Moorovo sousedství (von Neumann včetně diagonálně umístěných sousedů) nebo hexagonální sousedství (všechny buňky, které s aktuální buňkou sdílí hranu) v hexagonální mřížce. [16]

Vzdálenost, do které buňky v sousedství leží od středové buňky, se nazývá rádius. Typicky se velikost rádia řeší v 1D CA sousedstvích, ale existují i rozšířené varianty von Neumannova nebo Moorova sousedství, které počítají s rádiem $r > 1$ (viz obrázek 2.3). [38]



(a) (b)

Obrázek 2.3: Příklady 2D sousedství pro $r = 2$, a) von Neumannovo, b) Moorovo

Přechodová funkce

Uvažujme 2D celulární automat c a von Neumannovo sousedství. Stav buňky c s indexem i, j v diskrétním čase t označme jako $c_{i,j}(t)$. Pak je nový stav této buňky určen podle přechodové

funkce F na základě stavů buněk v sousedství, které leží do vzdálenosti r (rádius), v čase $t-1$ [16]:

$$c_{i,j}(t) = F(c_{i\ -r,j}(t-1), c_{i\ -r+1,j}(t-1), ..., c_{i\ +r-1,j}(t-1), c_{i\ +r,j}(t-1), \\ c_{i,j-r}(t-1), c_{i,j-r+1}(t-1), ..., c_{i,j+r-1}(t-1), c_{i,j+r}(t-1)).$$

Jedná se tedy o sousedství o velikosti $4r+1$. Každá buňka může nabývat právě jednoho z k možných stavů, $c_{i,j}(t) \in \{0, 1, 2, ..., k\}$ a přechodová funkce F je úplně definovaná pro každou možnou kombinaci stavů všech sousedů, což je celkem k^{4r+1} možných konfigurací sousedství. Výsledek (nový stav aktuální buňky) přechodové funkce F může být kterýkoliv z k možných stavů, a tudíž je celkem $k^{k^{4r+1}}$ možných pravidel. Jedná se tak o exponenciálně rostoucí funkci vůči počtu možných stavů k i rádiu r. Pro $r = 1$ (nerozšířené von Neumannovo sousedství) a $k = 2$ (buňka nabývá pouze hodnot 0, nebo 1) je to $2^{2^5} \approx 4,29 \cdot 10^9$ možných pravidel.

Přechodovou funkci je možno reprezentovat pomocí tabulky. Řádky tabulky odpovídají jednotlivým přechodovým pravidlům, která jsou složená z konkrétní konfigurace sousedství (hodnoty stavů buněk v sousedství) a hodnoty nového stavu středové buňky. Uvažujme přechodovou funkci F v 2D CA s von Neumannovým sousedstvím (s rádiem $r = 1$) a počtem možných stavů $k = 2$, $c_{i,j}(t) \in \{0, 1\}$. Nechť přechodová funkce F určí novou hodnotu stavu středové buňky $c_{i,j}(t)$ v sousedství jako 1, pokud jsou alespoň 3 buňky v sousedství (včetně středové buňky) s hodnotou stavu 1, jinak ji určí jako 0. Přechodová funkce F je reprezentována tabulkou 2.1. V tabulce jsou pro zjednodušení vypsána pouze ta přechodová pravidla, která vedou na změnu hodnoty stavu středové buňky.

Tabulka 2.1: Příklad přechodové funkce 2D CA s von Neumannovým sousedstvím

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
1	1	0	1	0	1
1	1	0	0	1	1
1	0	0	1	1	1
0	1	0	1	1	1
1	1	0	1	1	1
0	0	1	0	0	0
1	0	1	0	0	0
0	1	1	0	0	0
0	0	1	1	0	0
0	0	1	0	1	0

Problém může nastat už při hledání optimální sady pravidel, díky které by CA vykazoval požadované chování. Navíc jsou překážkou i velké paměťové nároky pro uchování pravidel při zpracování běžného 8bitového černobílého obrázku, kdy je pro takto velké množství stavů (256) využíváno von Neumannovo sousedství. Kdyby bylo jedno pravidlo zakódováno do 1 B, pak by jedna kompletní sada pravidel zabírala přes 1 TB a aktivní práce s takovými pravidly by byla prakticky nemožná [8]. Alternativami může být nastavení limitu počtu pravidel, totalistický CA[1] nebo podmínková pravidla (viz podkapitola 2.3).

[1] v totalistickém celulárním automatu [44] jsou stavy buněk reprezentovány jako hodnoty a výsledkem přechodové funkce je součet (průměr, medián, apod.) hodnot sousedních buněk.

2.2 Vybrané nejznámější varianty CA

V této podkapitole si uvedeme několik známých variant CA, *Game of life*, von Neumannův CA a Langtonovy smyčky. Tyto příklady nám ilustrují schopnost CA modelovat různé děje a vlastnosti.

Game of life

Jedná se o variantu 2D totalistického CA, jejíž autorem je John H. Conway, která využívá Moorova sousedství a dvou stavů – buňka je buď živá, nebo mrtvá [4]. V počátečním nastavení je buňkám náhodně určeno, zda jsou živé nebo mrtvé, a následně se synchronně vykonávají následující pravidla:

- Narození: Pokud je buňka mrtvá a právě 3 sousední jsou živé, pak je nahrazena živou buňkou.
- Smrt: Pokud je buňka živá a má nejvýše 1 živého souseda (smrt z izolace) nebo více než 3 živé sousedy (smrt z přeplněnosti), pak je nahrazena mrtvou buňkou.
- Přežití: Pokud je buňka živá a má 2 nebo 3 živé sousedy, pak zůstává živou.

Von Neumannův CA

John von Neumann přišel s myšlenkou systému, který měl schopnost vytvářet kopie sebe samého [27]. Společně s Stanislawem Ulamem tak položili základ diskrétních, 2D celulárních automatů. Na rozdíl od varianty Game of life, kde mohla být buňka buď živá, nebo mrtvá, měl von Neumannův CA 29 možných různých stavů. Dále v něm uplatnil von Neumannovo sousedství a množství komplikovaných přechodových pravidel různých typů (přenosová, slučující, konstrukční, destrukční).

Von Neumann využil tohoto automatu k sestrojení univerzálního konstruktoru, kde část buněk byla určena k uložení informace o popisu stroje a sloužila jako sada instrukcí pro vytvoření stroje. Stroj měl také dedikovanou část, která byla schopna tyto instrukce číst a provádět na volném místě v mřížce, aniž by původní instrukce mazala. Konstruktor tak při sebe-replikaci postaví libovolný konečný stroj na základě jeho popisu, a zároveň vytvoří i vlastní kopii na základě svého popisu. Nakonec je nutné nově vytvořené kopii poskytnou oba popisy, aby byl princip replikace kompletní.

Von Neumannův CA a konstruktor byly velkým průlomem a podnětem dalšího bádání a vylepšování. Zejména bylo snahou snížit počet stavů a zachovat schopnost sebe-replikace. Již Edgar F. Codd přišel s verzí celulárního automatu (Coddův CA), která pracovala pouze s 8 stavy [11]. Signál se v jeho provedení přenáší v trubicích (tunelech, drátech), jehož stěny určují směr jednotlivých signálů. Na konci trubice (pokračování do prázdna) se provede instrukce daná typem signálu a konec trubice se odpovídajícím způsobem prodlouží, zkrátí nebo jinak upraví. Velikost takového stroje byla ale příliš velká (283 milionů buněk).

Langtonovy smyčky

Úprava von Neumannova CA Christopherem Langtonem, Langtonovy smyčky, ukázala, že odstraněním vlastnosti univerzality lze výrazně snížit komplexitu celulárního automatu [22]. Podobně jako v Coddově CA se využívají trubice, ve kterých se přenáší signály a trubice se podle typu signálu dále modifikuje. Rozdíl je však v tom, že jediné, co je potřeba zachovat

je schopnost sebe-replikace – tedy se periodicky emituje jediná posloupnost signálů, které postupně rozšiřují trubici do odpovídajícího tvaru. Jakmile je stroj zreplikován (trubice je kompletní), speciální signály trubici zakončí a vytvoří začátek nové na vedlejším volném místě. Ukončená trubice již nemá schopnost sebe-replikace a je označena za mrtvou, nové trubice však pokračují v sebe-replikaci. Replikace je možno dosáhnout už za 151 kroků při počátečním počtu 86 buněk.

Další varianty CA

Porušením některých ze základních vlastností CA můžeme získat jiné zajímavé varianty jako třeba asynchronní CA, nehomogenní CA (přechodová pravidla nemusí platit pro všechny buňky), nedeterministické CA (přechodová pravidla jsou nahrazena pravděpodobnostmi výsledných stavů) nebo spojité CA (používají reálná čísla jako hodnoty stavů) [16]. Existují také CA, které umožňují dynamickou modifikaci mřížky nebo zapojení mřížky do evoluce tak, že se projevuje jako samostatná aktivní komponenta.

2.3 Podmínková přechodová pravidla

Smyslem podmínkových přechodových pravidel (CMR, z angl. *Conditionally Matching Rules*) je snížení počtu celkového počtu přechodových pravidel, aby bylo v jednom podmínkovém pravidle zahrnuto více „nepodmínkových“ pravidel [8].

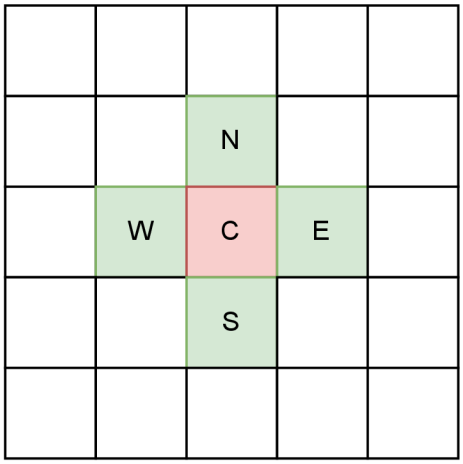
Mějme 2D CA s von Neumannovým sousedstvím s rádiem $r = 1$, pak podmínkové přechodové pravidlo má tvar:

$$(cnd_N\ s_N)(cnd_W\ s_W)(cnd_C\ s_C)(cnd_E\ s_E)(cnd_S\ s_S) \rightarrow s_C^{new},$$

kde *cnd* je podmínkový operátor, s je hodnota stavu buňky a identifikátory N, W, C, E a S odpovídají severní, západní, středové, východní a jižní sousední buňce, jak je naznačeno na obrázku 2.4. Podmínkové přechodové pravidlo se tedy skládá z levé části obsahující 5 podmínek a hodnot, které se porovnají se stavem buňky v sousedství, a pravé části s výslednou novou hodnotou buňky C. Mezi vhodné sady podmínkových operátorů a hodnot patří kombinace $c = 0$, $c \neq 0$, $c \geq s$ a $c \leq s$, kde c je hodnota stavu konkrétní buňky v sousedství a s je hodnotou určenou z intervalu $\langle 0, 255 \rangle$. Konečný počet těchto pravidel dává přechodovou funkci CA. Podmínková přechodová pravidla byla prozkoumána v několika publikacích, [6] a [7], a osvědčila se při filtrování šumu z digitálního obrazu v [8].

Uvažujme pravidlo tvaru „$\geq 150 \quad = 0 \quad \neq 0 \quad \leq 54 \quad = 0 \rightarrow 211$“ a hodnoty severní, západní, středové, východní a jižní buňky ze sousedství nechť nabývají hodnot 197, 0, 25, 31 a 0. Pak první podmínka pro severní sousední buňku je splněna, protože $197 \geq 150$. Druhá podmínka je také splněna, protože západní sousední buňka má hodnotu 0. Obdobně jsou splněny všechny ostatní podmínky a podmínkové pravidlo je tedy splněno. Novou hodnotou středové buňky bude tedy hodnota 211. Aby bylo pravidlo splněno, musí být splněny všechny podmínky na levé straně pravidla, jinak pravidlo splněno není a kontroluje se platnost jiného pravidla. Pokud jiné pravidlo nalezeno není, pak se hodnota středové buňky nemění.

Nechť je F přechodová funkce 2D CA s von Neumannovým sousedstvím složená z CMR. Pak přechodová funkce F postupně vyhodnocuje $l > 0$ CMR od prvního k poslednímu. Pokud je nalezeno vyhovující pravidlo, nová hodnota stavu středové buňky $c_{i,j}(t)$ je určena hodnotou na pravé straně pravidla a procházení pravidel se ukončí. V případě, že není nalezeno žádné vyhovující pravidlo, zůstává hodnota středové buňky nepozměněna, $c_{i,j}(t) =$



Obrázek 2.4: Označení sousedů ve von Neumannově sousedství, které bude uvažováno v této práci

$c_{i,j}(t-1)$. Přechodová funkce F může být také reprezentována tabulkou. V tabulce 2.2 je uvažován příklad přechodové funkce F s $l = 5$ CMR. Řádky odpovídají jednotlivým CMR, které jsou zapsané pomocí podmínkových operátorů a hodnot. Při určení nové hodnoty stavu středové buňky jsou pravidla vyhodnocována od prvního k poslednímu podle jejich pořadí v tabulce.

Tabulka 2.2: Příklad přechodové funkce 2D CA s von Neumannovým sousedstvím využívající CMR, buňky v sousedství jsou navíc označeny identifikátory N, W, C, E a S odpovídají severní, západní, středové, východní a jižní sousední buňce

$c_{i,j-1}(t-1)$ **(N)**	$c_{i\ -1,j}(t-1)$ **(W)**	$c_{i,j}(t-1)$ **(C)**	$c_{i\ +1,j}(t-1)$ **(E)**	$c_{i,j+1}(t-1)$ **(S)**	$c_{i\ ,j}(t)$ **(C)**
$= 0$	$\neq 0$	≥ 155	≥ 178	≤ 209	44
≥ 26	$= 0$	$\neq 0$	$= 0$	$\neq 0$	14
≤ 98	≥ 11	$\neq 0$	$= 0$	≤ 113	82
≤ 240	≤ 201	$= 0$	≥ 163	≤ 197	199
$\neq 0$	$= 0$	≥ 55	$= 0$	$\neq 0$	136

2.4 Použití CA pro filtraci obrazu

Filtraci šumu z obrazu lze řešit lineárními nebo nelineárními filtry [30]. Lineární filtry aplikují na každý pixel (a jeho okolí) matici váhových koeficientů, zatímco nelineární filtry obvykle hodnotu pixelu vypočítávají podle okolního hodnot. Mezi nelineární filtry patří například mediánové filtry nebo právě filtrace pomocí CA.

Konvenční metody

Konvenční přístup k eliminaci šumu z obrazu se skládá z detekce šumu a jeho filtrace. Algoritmus filtrace se zpravidla provádí pouze na pixelech nebo částech obrazu, které byly

identifikované jako poškozené. Jiné metody, například některé mediánové filtry, fázi detekce a filtrace slučují do vhodné vybrané filtrační funkce, jejíž vstupem jsou data obrazu a výstupem jsou filtrované pixely. Jednoduchý mediánový filtr (MF, nebo také SMF, z angl. *Simple Median Filter*) pro každý pixel počítá medián z celého filtrovacího posuvného okna (z angl. *filtering window*), které má obvykle velikost 3x3, nebo 5x5. Využití mediánu jako filtrační funkce je snadné a efektivní. Přesto nemusí být ideální, protože obraz filtrovaný jednoduchým mediánovým filtrem obvykle postrádá původní detaily a je příliš rozostřený. Existují však mnohé varianty, které se snaží původní koncept MF zdokonalit. Přehled a porovnání mediánových filtrů jsou shrnuty ve studii [37], přičemž byly vybrány dva z nejkvalitnějších filtrů – IDBMF (z angl. *Improved Decision Based Median Filter*) a EMF (z angl. *Extended Median Filter*).

Metoda IDBMF [2] se od běžných mediánových filtrů odlišuje tím, že využívá filtračního okna tvaru „+“, tedy v podstatě von Neumannova sousedství. K filtraci dochází pouze v případě, že je hodnota středového pixelu filtračního okna rovna 0 nebo 255, jinak je ponechána původní hodnota. Pokud mají všechny pixely filtračního okna hodnotu 0 nebo 255, filtrovanému pixelu je přiřazen průměr ze všech hodnot pixelů filtračního okna. Jinak mu je přiřazen medián z těch sousedních pixelů, které nenabývají hodnot 0 nebo 255.

Metoda EMF [10] pracuje s filtračním oknem 3x3 a při výpočtu mediánu nepočítá s hodnotou středového pixelu filtračního okna. Pokud je rozdíl hodnoty středového pixelu a mediánu větší než zvolená konstanta (v publikaci [10] je zvolena konstanta 20), pak je hodnotou filtrovaného pixelu medián, jinak je ponechána původní hodnota.

Metody využívající CA

Celulární automaty lze také využít k filtrování šumu z digitálního obrazu. Obraz o velikosti m x n pixelů můžeme uvažovat jako mřížku, pixely jako buňky a hodnoty pixelů jako jednotlivé možné stavy. Barevný obraz představuje příliš velkou množinu možných stavů, proto se nejčastěji uvažuje černobílý obraz buď s 256 odstíny šedi nebo binární obraz (obsahuje pouze černé nebo bílé pixely). Máme tedy základní nastavení 2D CA, ale to nejdůležitější je určení přechodové funkce, která by prováděla redukci šumu.

Bylo publikováno mnoho studií, které se zaobíraly různými přístupy k filtrování šumu za pomoci celulárních automatů. Mezi prvními navrhovanými strategiemi bylo porovnání hodnoty středové buňky vůči minimu, maximu a průměru hodnot sousedních buněk [23]. Pokud byla hodnota menší než minimum, větší než maximum nebo se příliš lišila od průměru, tak byla nahrazena odpovídající hodnotou. Dalším z přístupů je uplatnění majoritního pravidla [36]. Pokud má středový pixel hodnotu 0 nebo 255 (černý nebo bílý, potenciální šum), pak je mu nastavena hodnota, která je v jeho sousedství nejvíce zastoupena. V případě, že žádná hodnota v sousedství není majoritní, je buď fixně určeno, kterou sousední hodnotu vybrat, nebo je určena náhodná sousední hodnota. Dále je to nahrazení hodnotou danou průměrem sousedních hodnot, které nejsou 0 nebo 255 [40], nebo jedním z předem vypočítaných lokálních průměrů, který je vybrán na základě hodnot stavů sousedních buněk [29]. Relativně novým způsobem, jak filtrovat šum pomocí celulárních automatů, je i využití fuzzy logiky v přechodových pravidlech [34]. Nakonec metoda, na které je tato práce založená, využívá CMR tak, jak je naznačeno v kapitole 2.3 [8].

Obzvlášť výše zmíněna varianta CA, počítající s průměrem z nepoškozených hodnot v Moorově sousedství [40], dává kvalitní výsledky i při vysokých šumech typu sůl a pepř či *impulse burst*. Vzhledem k jednoduchosti i výpočetní nenáročnosti (algoritmus končí po

$(p/10) + 1$ krocích, kde p je procento šumu obrazu) budeme pro účely této práce považovat tuto metodu za modelovou.

Kapitola 3

Evoluční algoritmy

Tato kapitola je věnována evolučním algoritmům, konkrétně evolučním strategiím. V úvodní podkapitole jsou shrnuty principy evolučních algoritmů a jejich historie, poté následují evoluční strategie. V evolučních strategiích jsou rozebrány činnost algoritmu, typy strategií, reprezentace řešení, variační operátory a je nahlédnuto i do samo-adaptace řídicích parametrů variačních operátorů. Na konci kapitoly je shrnuto využití evolučních algoritmů k návrhu obrazových filtrů.

3.1 Úvod do evolučních algoritmů

Evoluční algoritmy (EA) [12] jsou stochastické metody globální optimalizace inspirované přírodou. Cílem těchto algoritmů je nalézt optimální řešení problémů, které mají více než jedno lokální optimum, pomocí technik založených na Darwinově teorii evoluce. Evoluční algoritmy se řadí mezi metaheuristiky[1] [24], pro které je typické hledání kompromisu mezi mírou explorace a exploatace[2]. Naopak společným rysem evolučních algoritmů je udržování si množiny kandidátních řešení, populace, proto jsou také někdy nazývané jako populační metody. Populace se může měnit s využitím variačních operátorů (mutace, křížení, aj.) a uplatňuje princip selekčního tlaku, tedy zvýhodnění (nebo naopak znevýhodnění) jedinců na základě konkrétních vlastností [45]. Populace by se tak měla postupně přibližovat ke kvalitnějším řešením, jejichž ohodnocení je dáno funkcí fitness. Výběr vhodné fitness funkce může ovlivnit, jak rychle a zda vůbec bude dosaženo optimálního řešení. Počet kroků evoluce závisí na ukončovací podmínce, zpravidla nalezení dostatečně kvalitního řešení nebo dosažení maximálního počtu generací. Princip činnosti jednoduchého evolučního algoritmu je naznačen v algoritmu 1.

Darwinovy principy inspirovaly mnohé vědce ke snaze je zautomatizovat nebo jinak zužitkovat při řešení velmi náročných problémů. Počátkem těchto snah byl návrh Alana Turinga využít přírodní evoluce v rámci umělé inteligence [42]. Nezávisle na sobě pak v 60. letech vzniklo několik různých realizací evolučních algoritmů, evoluční programování Lawrence J. Fogela, genetické algoritmy Johna H. Hollanda a evoluční strategie Inga Rechenberga a Hans-Paul Schwefela. Začátkem 90. let pak přišel John Koza s myšlenkou genetického programování. Zmíněná čtyři odvětví, evoluční programování (EP), evoluční strategie (ES), genetické algoritmy (GA) a genetické programování (GP), jsou považována

[1] Metaheuristiky jsou obecná schémata algoritmů, jeho částí a parametrů, po jejichž konkrétní specifikaci (pro daný problém) a vhodném nastavení parametrů vznikne fungující optimalizační algoritmus.

[2] Explorace algoritmu označuje jeho schopnost objevování nových dobrých řešení. Exploatace algoritmu je schopnost vylepšovat získaná řešení.

Algoritmus 1 Základní evoluční algoritmus

```
Inicializuj populaci s náhodnými kandidátními řešeními
Ohodnoť každého kandidáta funkcí fitness
while Ukončovací podmínka není splněna do
    Vyber rodiče z populace
    Získej křížením rodičů potomky
    Proveď mutaci potomků
    Ohodnoť potomky funkcí fitness
    Proveď selekci jedinců do nové populace
end while
```

za čtyři základní varianty evolučních algoritmů. Liší se od sebe zejména v technických detailech jako je reprezentace kandidátních řešení, typ variačních operátorů, selekce rodičů a potomků apod.

3.2 Evoluční strategie

Evoluční strategie (ES) vznikly podobně jako ostatní varianty evolučních algoritmů za účelem získat optimální řešení náročného problému využitím Darwinových principů evoluce [21]. Konkrétně se dvojice německých vědců, Rechenberg a Schwefel, pokoušela nalézt takový tvar tělesa, který by měl co nejmenší součinitel odporu v aerodynamickém tunelu. Zaměřili se zejména na náhodné změny jednotlivých vlastností daného tvaru v průběhu evoluce. Jejich inovativní práce přinesla nové možnosti, jak přistupovat k problému hledání vhodného nastavení parametrů systému reprezentovaných mnohadimenzionálními vektory reálných čísel. V dnešní době patří ES mezi jedny z nejdůležitějších algoritmů uplatněných v optimalizaci funkcí, pro které není k dispozici analytické řešení.

Mezi charakteristiky, jimiž se ES odlišují od ostatních evolučních algoritmů, se řadí reprezentace jedinců odpovídající skutečnosti (typicky reálná čísla), náhodný výběr při volbě rodičovských kandidátních řešení, adaptivní parametrizace variačních operátorů a deterministická selekce potomků [33].

Algoritmus 2 Obecný algoritmus evoluční strategie

```
Inicializuj populaci s náhodnými kandidátními řešeními
Ohodnoť každého kandidáta funkcí fitness
while Ukončovací podmínka není splněna do
    Vyber náhodně rodiče s rovnoměrnou pravděpodobností
    Získej potomky aplikací variačních operátorů na rodiče
    Ohodnoť potomky funkcí fitness
    Seřaď potomky (a případně i rodiče) podle hodnoty fitness
    Proveď selekci nejlepších jedinců do nové populace
end while
```

Činnost algoritmu ES (viz algoritmus 2) zahrnuje většinu výše zmíněných vlastností. Z počáteční náhodně inicializované populace jsou rovnoměrně náhodně vybrána kandidátní řešení nazývaná rodiče. Z těch jsou pomocí variačních operátorů získána nová kandidátní řešení, neboli potomci, kteří jsou ohodnoceni fitness funkcí a seřazeni podle kvality řešení dané hodnotou fitness. Následně se provede deterministická selekce z potomků nebo i ro-

dičů podle typu zvolené strategie. Celý proces se opakuje, dokud není splněna ukončovací podmínka.

Forma selekce potomků je daná typem vybrané strategie, přičemž existují dvě základní: strategie „čárka“ (*the „comma“ strategy*) a strategie „plus“ (*the „plus“ strategy*) [21]. Nechť μ označuje velikost populace a λ počet vygenerovaných potomků v jedné iteraci, pak můžeme strategii „čárka“ označit jako (μ, λ)-ES a strategii „plus“ jako (μ + λ)-ES.

(μ, λ)-ES

V této strategii je původní populace zcela nahrazena μ nejkvalitnějšími vygenerovanými potomky. Životnost kandidátního řešení je tedy pouze 1 generace. Zahozením rodičovských kandidátních řešení se může algoritmus vyvarovat uváznutí v lokálním optimu, což je pro řešení multimodálních problémů[3] výhodné. Selektivní tlak je v této strategii vysoký, obzvlášť pokud je zvolena násobně vyšší λ oproti μ (typicky se volí poměr 1/7 nebo 1/4). Množství generovaných potomků tak řídí míru explorace algoritmu, zatímco velikost populace exploataci [24].

(μ + λ)-ES

Strategie „plus“ v selekci zohledňuje nejen potomky, ale i původní rodičovská kandidátní řešení. Přirozenou vlastností tohoto algoritmu je tedy automatický přenos dosud nejlepších nalezených řešení do nové populace a zvýhodnění těchto řešení oproti jiným. Tento jev se také označuje jako „elitismus“ [15]. Tento typ strategie s sebou nese větší exploataci. Nevýhodou těchto strategií je ale předčasná konvergence k lokálnímu optimu kolem kandidátního řešení, které by opakovaně vítězilo v selekci [24].

Reprezentace kandidátních řešení

Evoluční strategie se typicky využívají v doméně reálných čísel, $\mathbb{R}^n$. Jedinec $\bar{\mathbf{x}}$ je reprezentován jako vektor reálných čísel o délce n [33]. Některé varianty ES však pracují s binárními nebo celými čísly, například v úlohách zpracování obrazu, kde jednotlivé pixely nabývají celočíselných hodnot. Reprezentaci jedince nazýváme také jako chromozom. Uvažujme následující příklady chromozomů pro domény reálných, celých a binárních čísel:

$$\bar{\mathbf{x}}_{\mathbf{R}} = [1.26, -2.98, 98.06, ...] \text{ pro } \bar{\mathbf{x}}_{\mathbf{R}} \in \mathbb{R}^n,$$

$$\bar{\mathbf{x}}_{\mathbf{Z}} = [30, 105, -15698, ...] \text{ pro } \bar{\mathbf{x}}_{\mathbf{Z}} \in \mathbb{Z}^n,$$

$$\bar{\mathbf{x}}_{\mathbf{B}} = [0, 0, 1, 0, 1, ...] \text{ pro } \bar{\mathbf{x}}_{\mathbf{B}} \in \mathbb{B}^n.$$

Variační operátory

Mutace a křížení jsou dva základní variační operátory, které se v ES běžně využívají. Návrh vhodných variačních operátorů ES pro daný problém by se měl řídit několika pravidly [33]. Prvním pravidlem je dosažitelnost jakéhokoliv kandidátního řešení z počátečního řešení po konečném počtu kroků. Další zásadou je podobně jako při výběru rodičů nezaujatost. Aplikace variačních pravidel by neměla dávat přednost konkrétním skupinám řešení, přičemž

[3] V kontextu optimalizačních algoritmů jsou multimodální problémy takové funkce, které mají více než jedno lokální minimum (maximum).

malé změny jsou častější než ty velké. Posledním pravidlem je využití kontrolních parametrů k ovládání variačních operátorů.

Velký význam je přikládán mutaci, která je hlavním zdrojem variace a v ES je vždy přítomna. Některé ES zcela vynechávají křížení a pro generování potomků využívají opakovanou mutaci rodičovských řešení. V závislosti na reprezentaci jedince se obvykle volí mutace s normálním rozdělením pro hodnoty z $\mathbb{R}^n$, s geometrickým rozdělení pro hodnoty z $\mathbb{Z}^n$ a diskrétním rovnoměrným rozdělením pro hodnoty z $\mathbb{B}^n$ [33]. Takové mutace zajišťují maximální entropii, zjednodušeně maximální míru neuspořádanosti systému. Mutace s rovnoměrným rozdělením je vhodná v situaci, kdy máme jen určitý interval povolených reálných hodnot. Pro celočíselné reprezentace se nabízí jednoduchý náhodný výběr libovolné z povolených hodnot [12]. Jedním z parametrů, který ovlivňuje míru mutace je tzv. mutační krok, obvykle značený σ. Čím vyšší je jeho hodnota, tím mocnější mutace. Uveďme si příklad způsobu mutace pro reálná čísla:

$$x' = x + \mathcal{N}(0, \sigma), \quad x, x' \in \bar{\mathbf{x}}$$

$$\sigma = \frac{x_{max} - x_{min}}{6}$$

a příklad pro celá čísla pro problém mutace nad 8bitovým černobílým obrazem (hodnoty x náleží do $\{0...255\}$):

$$x' = \mathcal{U}(0, 255), \quad x' \in \bar{\mathbf{x}},$$

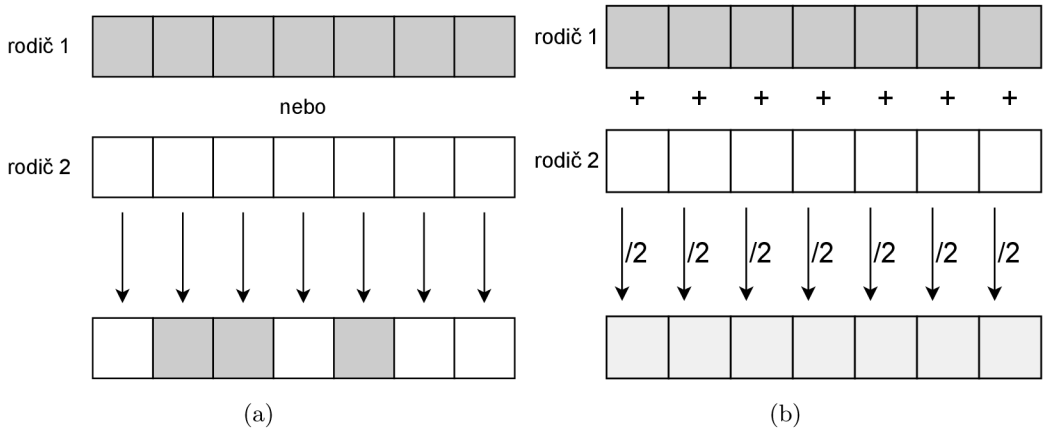
kde x' je nová hodnota získaná mutací x. V prvním příkladu je hodnota prvku jedince $\bar{\mathbf{x}}$ zmenšena (nebo zvětšena) o hodnotu získanou z normálního rozdělení daného mutačním krokem. Ten je určen jako konstantní hodnota na základně intervalu povolených hodnot. U druhého z uvedených příkladů není původní hodnota braná v potaz a nová hodnota je vybrána zcela náhodně z intervalu $\langle 0, 255 \rangle$. Mutační krok by v tomto případě mohl označovat počet mutací v chromozomu jedince, tedy počet náhodně zvolených prvků vektoru $\bar{\mathbf{x}}$, jejichž hodnoty budou změněny. Uvažujme pro tento případ mutační krok $\sigma = 3$ a dva možné nové vektory $\bar{\mathbf{x}}'_\mathbf{1}$ a $\bar{\mathbf{x}}'_\mathbf{2}$ získané mutací $\bar{\mathbf{x}}$:

$$\bar{\mathbf{x}} = [0, 153, 69, 87, 236], \quad \bar{\mathbf{x}}'_\mathbf{1} = [0, \underline{2}, \underline{198}, 87, \underline{99}], \quad \bar{\mathbf{x}}'_\mathbf{2} = [\underline{47}, 153, \underline{255}, 87, \underline{34}].$$

Dalším variačním operátorem je křížení. V ES se obvykle používají dvě metody křížení, diskrétní křížení a křížení průměrem [12]. Při diskrétním křížení jsou vybírány jednotlivé hodnoty vektoru z těch rodičovských náhodně, přičemž všichni rodiče mají stejnou šanci výběru. Křížení průměrem pak získává jednotlivé hodnoty zprůměrováním všech hodnot rodičů. Na obrázku 3.1 jsou ilustrovány oba typy křížení. Standardně se křížení účastní dva rodiče, tzv. lokální křížení, ale existuje i varianta pro více rodičů, globální křížení. Některé algoritmy pro každou hodnotu vektoru potomka provádí pokaždé nový výběr rodičů nebo používají různé techniky křížení zvlášť pro vektor jedince a vektor kontrolních parametrů.

Samo-adaptace řídicích parametrů variačních operátorů

Reprezentace jedince může být rozšířena o další hodnotu nebo vektor hodnot s kontrolními parametry [9]. Chromozom tak neobsahuje pouze hodnoty řešení problému, ale i hodnoty, které se týkají specifického nastavení evoluce. S parametrizací variačních operátorů je spojen koncept samo-adaptace řídicích parametrů variačních operátorů. Tyto řídicí parametry z rozšířené reprezentace jedince kontrolují průběh evoluce a mohou se specifickým způsobem během evoluce měnit. Chromozom tak bude mít nově tvar $\langle \bar{\mathbf{x}}, \sigma \rangle$, kde $\bar{\mathbf{x}}$ je původní vektor a σ je řídicí parametr, v tomto případě mutační krok.



Obrázek 3.1: Nejčastější typy křížení v ES, a) diskrétní křížení, b) křížení průměrem

Z prvních experimentů nad ES vyplynulo, že je potřeba zavést kontrolu velikosti kroku mutace. Cílem je provádět menší změny, čím blíže je evoluce k optimálnímu řešení. Jednou z nejznámějších metod je pravidlo pětiny úspěšných mutantů (1/5-*success rule*) [33], která počítá počet úspěšných mutací s za určitý počet generací p. Pokud je podíl $s\ /p$ menší než 1/5, znamená to, že algoritmus příliš prozkoumává okolí, pak je třeba snížit mutační krok σ. Jestliže je podíl větší, tak se naopak neustále pohybujeme blízko lokálního optima a mutační krok zvýšíme. Jinak se mutační krok nemění. Pro změnu mutačního kroku je doporučena hodnota $c = (20/17)^{1/n}$. Formálně lze pak adaptaci mutačního kroku zapsat takto:

$$\sigma = \begin{cases} \sigma/c & \text{pokud } s\ /p < 1/5 \\ \sigma \cdot c & \text{pokud } s\ /p > 1/5 \\ \sigma & \text{pokud } s\ /p = 1/5 \end{cases}$$

3.3 Evoluční návrh obrazových filtrů

Návrh vhodných nelineárních obrazových filtrů není triviální. V poslední době se proto objevují přístupy využívající EA k návrhu těchto filtrů, zejména v oblasti evolučního hardware (EHW, nebo jen EH, z angl. *Evolvable Hardware*). Evoluční hardware [41] je odvětví zaměřující se na využití EA pro vytvoření adaptivních zařízení, která se dokážou bez manuálního zásahu přizpůsobovat dynamicky měnícímu se prostředí. Jedním z hlavních představitelů v této oblasti je kartézské genetické programování (CGP, z angl. *Cartesian Genetic Programming*) [35]. CGP využívá (1 + λ)-ES bez křížení (pouze mutace) [26] k návrhu matice (digitální obvody, programovatelná hradlová pole) složené z nízkoúrovňových výpočetních jednotek řešících odstranění šumu z obrazu s velmi nízkými výpočetními nároky. Výsledky z rozsáhle studie na téma CGP [43] budou uvažovány jako další modelová metoda pro porovnání s výsledky této práce.

Mezi další příklady využití EA k návrhu obrazových filtrů patří metoda z publikace [18] zabývající se gramatikou-řízeným genetickým programováním (GGGP, z angl. *Grammar-Guided Genetic Programming*) nebo metoda z publikace [3], která pomocí genetického algoritmu (GA) optimalizuje složitost obvodu provádějícího filtraci obrazu. Metoda z publikace [39] opět využívá CGP pro filtraci obrazu, ale v kombinaci v genetickým programová-

ním (GP) pro výpočet funkce fitness, kdy se spolu v koevoluci paralelně vyvíjí dvě nezávislé populace (jedna pro CGP, jedna pro GP). Jako poslední zde bude zahrnuta již zmíněná metoda (viz podkapitola 2.4) využívající (4, 8)-ES s elitismem pro návrh přechodové funkce CA složené z CMR [8].

Kapitola 4

Evoluce obrazových filtrů využívajících CA

V této kapitole je detailně vysvětlen princip navrhované metody (využití CA, modifikace CMR, funkce fitness, varianta a nastavení ES) a uvažované modely šumu včetně typů poškození využitých v experimentální části práce.

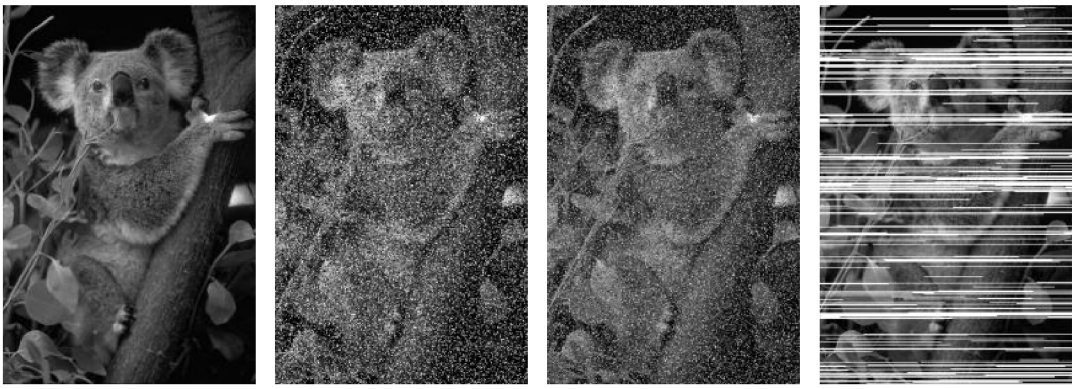
4.1 Uvažované modely šumu

Přestože jsou dnešní systémy velmi pokročilé, stále nejsou zcela dokonalé a k poškození obrazu může dojít při jeho pořizování, přenosu nebo ukládání. Obraz může být poškozen různými způsoby, pro navrhovanou metodu budeme uvažovat poškození šumem sůl a pepř (z angl. *salt and pepper noise*), *impulse burst* a náhodným šumem (z angl. *random shot noise*). Všechny zmíněné šumy poškozují obraz záměnou hodnot pixelů. V obraze poškozeném šumem typu sůl a pepř nabývají náhodně poškozené pixely pouze minimální nebo maximální intenzity pixelu, tedy hodnoty 0 nebo 255. Intenzitu šumu sůl a pepř v obraze I o velikosti m x n pak můžeme vyjádřit jako pravděpodobnost P, že bude pixel poškozen jako:

$$I_{i,j} = \begin{cases} 0 & \text{s pravděpodobností } P/2 \\ 255 & \text{s pravděpodobností } P/2 \\ I_{i,j} & \text{s pravděpodobností } 1 - P, \end{cases}$$

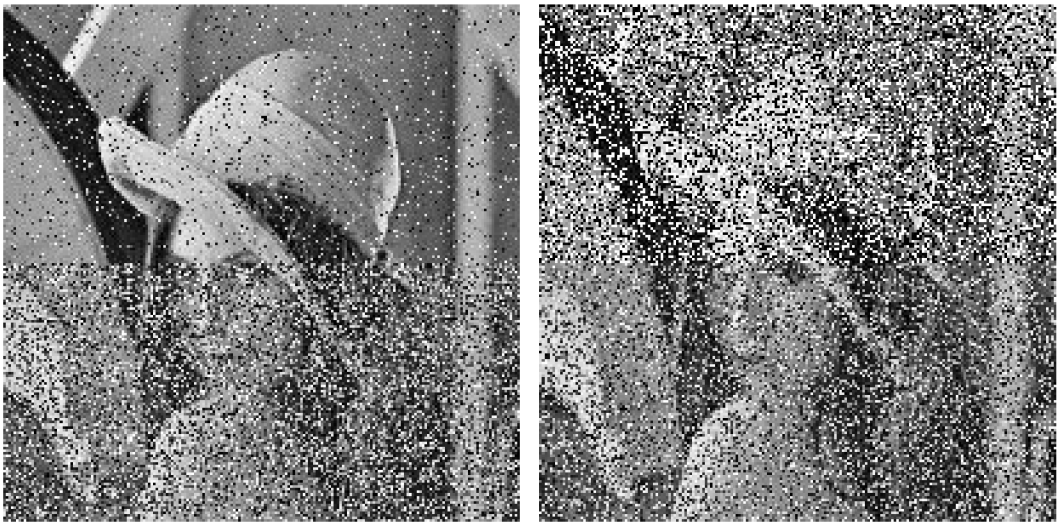
kde $I_{i,j}$ je hodnota pixelu obrazu I na souřadnicích i a j, kde $0 \geq i \geq m$ a $0 \geq j \geq n$. U náhodného šumu může být poškozený pixel nahrazen jakoukoliv z možných hodnot s rovnoměrnou pravděpodobností $\mathcal{U}(0, 255)$. Šum *impulse burst* poškozuje náhodně dlouhé posloupnosti pixelů v obraze maximální hodnotou (255). Vizuálně se jedná o vodorovné (nebo svislé) bílé pásy v obraze. Na obrázcích 4.1 (b–d) jsou předvedeny jednotlivé šumy při intenzitě šumu 30 %.

Kromě standardního postupu, kdy je celý obraz poškozen daným typem šumu ve vybrané intenzitě, byl během evolučního trénování filtru (návrhu pravidel celulárního automatu) uvažován i alternativní přístup zahrnující poškození každé poloviny obrazu jinou intenzitou (včetně 0% intenzity – ponechání poloviny obrazu zcela bez poškození) i různými typy šumu. Příklady takto poškozeného obrazu jsou zobrazeny na obrázcích 4.2 (a, b).



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

Obrázek 4.1: 8-bitový obraz 69015 (koala), 321x481px, a) bez poškození, b) poškozený šumem sůl a pepř s intenzitou 30 %, c) poškozený náhodným šumem s intenzitou 30 %, d) poškozený šumem *impulse burst* s intenzitou 30 %. Původní nepoškozený obraz je součástí datové sady univerzity Berkeley, k dispozici na `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/gray/test-026-050.html`



(a) (b)

Obrázek 4.2: 8-bitový obraz Lena, 200x200px, a) poškozený náhodným šumem s intenzitou 10 % na horní polovině obrazu a 60 % na spodní polovině obrazu %, b) poškozený šumem sůl a pepř s intenzitou 50 % na horní polovině obrazu a náhodným šumem s intenzitou 50 % na spodní polovině obrazu. Původní nepoškozený obraz byl převzat z: `https://www.cosy.sbg.ac.at/~pmeerw/Watermarking/lena.html`

4.2 Filtrace šumu pomocí CA

Využití konceptu CA pro filtraci obrazu, kterým se zabývá tato práce, bude uvažováno v následujícím uspořádání. Počáteční stav CA reprezentuje hodnoty pixelů poškozeného

obrazu. Cílem je nalézt taková pravidla CA, kterými po daném počtu kroků CA zrekonstruuje v co nejlepší kvalitě původní obraz (tj. odstraní poškozené pixely). Bude se jednat o 2D CA využívající von Neumannovo sousedství s přechodovou funkcí, která je složena z CMR s modifikací pravé strany pravidla.

4.3 Reprezentace pravidel CA

Navrhovaná metoda filtrace pomocí CA využívá CMR, která jsou blíže popsána v kapitole 2.3, s modifikací pravé strany pravidla. Motivací k úpravě pravé strany CMR je omezený počet možných nových hodnot stavů, což se ukazuje jako limitující u vyšších intenzit šumu. Zavedení výpočtu mediánu, průměrů, nebo jiných funkcí závislých na hodnotách stavů buněk v sousedství se navíc jeví jako žádoucí s přihlédnutím k efektivitě metod, které tyto funkce k filtraci obrazu používají (viz kapitola 2.4).

Modifikace pravé strany CMR spočívá v zavedení vhodné výpočetní funkce místo konstantní hodnoty:

$$(cnd_N\ s_N)(cnd_W\ s_W)(cnd_C\ s_C)(cnd_E\ s_E)(cnd_S\ s_S) \rightarrow func(neighs),$$

kde $func$ je zvolená výpočetní funkce, jejíž vstupními parametry jsou hodnoty stavů buněk v sousedství, $neighs = (N, W, C, E, S)$, a výstupní hodnotou je nová hodnota stavu středové buňky C. Mezi funkce prozkoumané v této práci byly zařazeny medián ($median$), průměr ($mean$), Gaussův filtr pro $\sigma = 1$ ($gauss$), bilaterální filtr pro $\sigma_s = 1$ a $\sigma_r = 0,1 \cdot 255 = 25.5$ ($bilateral$), minimum (min), maximum (max), majoritní hodnota (nejvíce zastoupená hodnota, $major$), minoritní hodnota (nejméně zastoupená hodnota, $minor$), polovina minima a maxima ($half$), zvolení hodnoty stavu buňky v sousedství (N, W, E nebo S) a konstantní hodnota získaná evolucí (původní s_C^{new}, nově jako val).

Uvažujme pravidlo tvaru „$\geq 150\ \ = 0\ \ \neq 0\ \ \leq 54\ \ = 0 \rightarrow major$“ a hodnoty severní, západní, středové, východní a jižní buňky ze sousedství nechť nabývají hodnot 197, 0, 25, 31 a 0. Levá strana pravidla i hodnoty stavů buněk v sousedství jsou totožné s příkladem z podkapitoly 2.3, tedy všechny podmínky jsou splněny. Novou hodnotou stavu středové buňky C je majoritní hodnota, nejvíce zastoupená hodnota mezi hodnotami stavů buněk v sousedství, kterou je v tomto příkladě hodnota 0. Dále uveďme příklad přechodové funkce F reprezentované tabulkou 4.1. Způsob procházení pravidel zůstává stejný jako v původním konceptu CMR.

Tabulka 4.1: Příklad přechodové funkce 2D CA s von Neumannovým sousedstvím využívající CMR s modifikací pravé strany pravidla, buňky v sousedství jsou označeny identifikátory N, W, C, E a S odpovídají severní, západní, středové, východní a jižní sousední buňce

$c_{i,j-1}(t-1)$ **(N)**	$c_{i\ -1,j}(t-1)$ **(W)**	$c_{i,j}(t-1)$ **(C)**	$c_{i\ +1,j}(t-1)$ **(E)**	$c_{i,j+1}(t-1)$ **(S)**	$c_{i\ ,j}(t)$ **(C)**
$= 0$	$\neq 0$	≥ 155	≥ 178	≤ 209	$median$
≥ 26	$= 0$	$\neq 0$	$= 0$	$\neq 0$	14
≤ 98	≥ 11	$\neq 0$	$= 0$	≤ 113	min
≤ 240	≤ 201	$= 0$	≥ 163	≤ 197	$mean$
$\neq 0$	$= 0$	≥ 55	$= 0$	$\neq 0$	N

Zavedení více různých možností na pravou stranu pravidla přináší poměrně značné zvětšení prohledávacího prostoru. Je tedy žádoucí snížit počet možných funkcí a získat tak

vhodnou sadu výpočetních funkcí dodávající nejkvalitnější filtraci obrazu v důsledku aplikace pravidel CA. Tento problém je dále prozkoumán v experimentální části práce (viz podkapitola 5.2).

4.4 Fitness funkce

Funkce fitness spočívá v provedení několika kroků filtrace CA a výpočtu střední kvadratické odchylky (MSE, z angl. *Mean Squared Error*). Nechť máme dva 8bitové černobílé obrazy C a R o velikost m x n pixelů a C odpovídá poškozenému obrazu, na který byl aplikován filtr CA, a R je původní nepoškozený obraz. Pak MSE je definována takto:

$$\text{MSE} = \frac{1}{m \cdot n} * \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (C(i,j) - R(i,j))^2.$$

Pro účely srovnání kvality různých filtrů je výhodnější použít ukazatel PSNR (špičkový poměr signálu k šumu, z angl. *peak signal-to-noise-ratio*). Nechť maximální hodnota pixelu v obrázku MAX_I je 255, pak rovnice pro odvození PSNR je:

$$\text{PSNR} = 20 \cdot \log_{10}(MAX_I) - 10 \cdot log_{10}(\text{MSE}).$$

Čím větší hodnota PSNR, tím kvalitnější je filtrovaný digitální obraz. Hodnocení obrazových filtru podle PSNR je standardní postup a je vhodný pro porovnání kvalit různých filtrů na stejných příkladech.

4.5 ES pro návrh obrazových filtrů

Podmínkové přechodové pravidlo lze reprezentovat jako posloupnost celých čísel o pevném počtu prvků. Levá strana pravidla se skládá ve von Neumannově sousedství z 5 párů tvořených podmínkovým operátorem a hodnotou, přičemž máme 4 možné podmínky ($=$, $\neq$, $\leq$ nebo $\geq$) a 256 možných hodnot. Pravá strana pravidla kóduje pouze vybranou výpočetní funkci nebo konkrétní hodnotu. Metoda uvažuje s variantami 4 a 8 výpočetních funkcí, které mohou být přiřazený na pravou stranu CMR. Jedno CMR tedy lze zakódovat do 11 celých čísel. Pro přechodovou funkci CA o počtu CMR R se jedná o posloupnost R x 11 celých čísel. Cílem metody je nalézt pomocí evoluce takovou přechodovou funkci, díky které bude v daném počtu kroků CA kvalitně rekonstruovat původní obraz. Metoda z článku [8] pracuje s $R = 20$ a $R = 30$, bude tedy žádoucí prozkoumat vliv různého počtu pravidel R na kvalitu filtrace obrazu.

V této práci budeme uvažovat scénář s konkrétní variantou (μ, λ)-ES, a sice (4, 8)-ES. Chromozom jedince reprezentuje výše zmíněná celočíselně zakódovaná přechodová funkce CA. Počáteční populace se skládá z náhodně vygenerovaných jedinců. Na začátku každé generace jsou všichni jedinci aktuální populace ohodnoceni funkcí fitness a seřazení od nejlepšího k nejhoršímu s indexem $i \in \{1, 2, 3, 4\}$. Z každého jedince jsou mutací vytvořeni dva potomci. Mutace jedince spočívá ve změně i čísel na náhodných místech chromozomu na novou hodnotu, která je vybrána z odpovídajícího intervalu možných hodnot s rovnoměrnou pravděpodobností. Mutační krok je tedy dán podle kvality jedince – kvalitnější jedinci jsou mutováni méně (mají nižší index i), zatímco méně kvalitní více. Celkem 8 vygenerovaných potomků je ohodnoceno a 4 nejkvalitnější z nich jsou selektováni do nové populace. Jedinci

z původní populace a 4 nejméně kvalitní potomci jsou zahozeni. Metoda navíc pracuje se zapojením elitismu, kdy se v populaci neustále udržuje dosud nejlepší nalezené řešení.

Ukončovací podmínkou je maximální počet generací, po jehož dosažení je výsledkem ES řešení s nejlepší fitness hodnotou.

Kapitola 5

Experimentální výsledky

Kapitola se zabývá nastavením experimentů a způsobem jejich vyhodnocení. Postupně jsou vysvětleny hlavní cíle experimentů, nalezení vhodné sady výpočetních funkcí pro pravou stranu modifikovaných CMR a nejlepší kombinace parametrů nastavení ES generující nejkvalitnější filtry pro vybrané modely šumu. Výsledky experimentů jsou vizualizovány v grafech a praktických ukázkách a jsou diskutovány. Nejlepší dosažené filtry jsou reprezentovány tabulkou a porovnány s existujícími metodami.

5.1 Experimentální nastavení

Celkem bylo provedeno 3234 experimentů s různými výběry možných výpočetních funkcí na pravé straně CMR a 5460 experimentů s jedním výběrem možných výpočetních funkcí. Každý experiment měl 30 nezávislých stochastických běhů. Jako trénovací obraz byl zvolen známý snímek s názvem Lena (viz obrázek 5.1) o velikosti 200x200 pixelů. Jako trénovací typy šumů byly zvoleny šum sůl a pepř, náhodný šum a jejich kombinace. Šum *impulse burst* nebyl do trénování zařazen, neboť se během experimentování ukázalo (viz podkapitola 5.3), že filtry trénované na šumu typu sůl a pepř odstraňují stejně dobře i poškození šumem *impulse burst*. Pro experimenty nad jedním výběrem možných výpočetních funkcí byly zvoleny tyto hodnoty parametrů:

- jedna intenzita šumu na celém obraze:
  - 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 %;
- rozdílné intenzity stejného typu šumu na polovinách obrazu (dvojice intenzit pro horní a spodní polovinu obrazu):
  - horní polovina obrazu je nepoškozena (intenzita šumu je 0 %), spodní polovina obrazu je poškozena šumem s intenzitou 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 %,
  - obě poloviny obrazu jsou poškozeny šumem s intenzitami 10 % a 60 %, 10 % a 90 %, 20 % a 60 %, 30 % a 60 %, 40 % a 60 %, 40 % a 80 %, 50 % a 70 %;
- rozdílné nebo stejné intenzity rozdílného typu šumu na polovinách obrazu (dvojice intenzit pro horní a spodní polovinu obrazu):
  - obě poloviny obrazu jsou poškozené rozdílnými šumy se stejnou intenzitou 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 %,

  - obě poloviny obrazu jsou poškozené rozdílnými šumy s rozdílnými intenzitami 10 % a 90 %, 90 % a 10 %, 20 % a 80 %, 80 % a 20 %, 40 % a 60 %, 60 % a 40 %;

- počet kroků filtrace CA:
  - 1, 2, 3, 4, 5, 6;
- počet CMR přechodové funkce CA:
  - 5, 10, 15, 20, 30, 40, 50;
- počet možných výpočetních funkcí na pravé straně CMR:
  - 4, 8.

Nastavení experimentu podle výše zmíněných parametrů je zakódováno do názvu experimentu následujícím způsobem:

$$\mathbf{n}\textit{salt\_pepper\_halved\_}\mathbf{d}\textit{60\_}\mathbf{t}\textit{median-min-max-val-N-W-E-S\_}\mathbf{s}\textit{3\_}\mathbf{r}\textit{10}, \tag{5.1}$$

kde tučně zvýrazněná písmena jsou následující zkratky:

- **n** pro *noise*, typ šumu;
- **d** pro *damage*, intenzita šumu;
- **t** pro *transition function*, výběr možných výpočetních funkcí pro pravou stranu CMR;
- **s** pro *steps*, počet kroků filtrace CA;
- **r** pro *rules*, počet CMR přechodové funkce CA.

V uvedeném příkladě 5.1 byl trénovací obraz poškozen šumem typu sůl a pepř s rozdílnými intenzitami na polovinách obrazu (*halved*). Horní polovina obrazu zůstala nepoškozena, spodní polovina obrazu byla poškozena šumem s intenzitou 60 %. Počet výpočetních funkcí, které je možné dosadit na pravou stranu CMR je 8 a jedná se o medián, minimum, maximum, evolvovanou konstantní hodnotu a hodnoty stavů buněk v sousedství (bez středové buňky). Počet kroků filtrace CA je 3 a počet CMR přechodové funkce CA je 10.

Obrázek 5.1: Obraz Lena, 200x200px

Pro vyhodnocení experimentů byl následně použit set 25 černobílých obrazů (viz obrázky 5.2(a-y)) veřejně poskytovaný univerzitou Berkeley[1] poškozený šumy typu sůl a pepř,

[1]K dispozici na `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/gray/test-026-050.html`

impulse burst a náhodným šumem o intenzitách 10, 20, 30, 40, 50, 60, 70, 80 a 90 %. Výsledné ohodnocení experimentů na daném typu šumu pak bylo vypočítáno jako průměrná hodnota PSNR pro každou intenzitu ze všech obrazů ze všech běhů experimentu.

Pro potřeby zhodnocení výsledků této práce byly implementovány všechny zmíněné konvenční metody z podkapitoly 2.4 a metody z článků [40] a [8]. Metodu z článku [43] bohužel nešlo snadno zreprodukovat, proto jsou alespoň přiloženy grafy a ukázky filtrace přímo z publikace na obrazech v ní použitých[2].

[2]Obrazy 317080 (srna), 189011 (chlapec), 299091 (pyramida), 106202 (tučňák) a 159008 (liščata) jsou také ze sady obrazů univerzity Berkeley. K dispozici na `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/gray/`

Obrázek 5.2: Obrazy použité k vyhodnocení jednotlivých experimentů. Převzato z: `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/gray/test-026-050.html`

5.2 Výběr sady výpočetních funkcí pravé strany CMR

Při hledání optimální sady výpočetních funkcí byly provedeny experimenty jak pro velikost sady 4, tak 8. Pro velikost sady 4 se hledaly experimenty dávající nejvyšší hodnotu PSNR s různými kombinacemi možných výpočetních funkcí se shodným nastavením náhodného šumu o intenzitě 50 %, počtu kroků filtrace CA 5 a počtu CMR přechodové funkce CA 20. Stejný postup byl zvolen pro velikost sady 8, kde bylo nastavení experimentů šum sůl a pepř s různou intenzitou na polovinách obrazu 0 % a 60 %, počet kroků CA 3 a počet CMR 10. Experimenty byly porovnávány vůči sobě v rámci sady stejné velikosti se stejným nastavením (mimo kombinace výpočetních funkcí), tedy na stejném obraze se stejným poškozením a počtem kroků filtrace, proto bylo vynecháno plné vyhodnocení experimentů.

Tabulka 5.1: 10 nejlepších experimentů podle mediánu PSNR pro sadu výpočetních funkcí velikosti 8, kde je každý experiment určen podle zahrnuté výpočetní funkce („T“ pro experimenty s danou funkcí, „F“ bez dané funkce).

	medián PSNR	*median*	*mean*	*min*	*max*	*bilateral*	*gauss*	*half*	*major*	*minor*	*N*	*S*	*W*	*E*	*val*
1	26.070	T	F	T	T	F	F	F	F	F	T	T	T	T	T
2	26.048	T	T	T	T	F	F	F	F	T	T	T	F	T	F
3	25.952	T	T	F	T	T	T	F	F	T	T	F	F	F	F
4	25.944	T	T	T	T	T	F	F	F	F	T	F	T	F	T
5	25.930	T	F	T	T	F	F	T	F	T	F	T	T	F	T
6	25.926	T	T	T	T	T	T	F	F	T	F	F	F	F	T
7	25.907	T	F	T	T	T	F	F	F	F	T	T	T	T	F
8	25.905	T	F	T	T	F	F	T	F	F	T	T	T	F	T
9	25.902	F	F	T	T	T	F	F	T	F	T	T	T	F	T
10	25.895	T	F	T	T	F	T	F	T	F	T	T	F	F	T

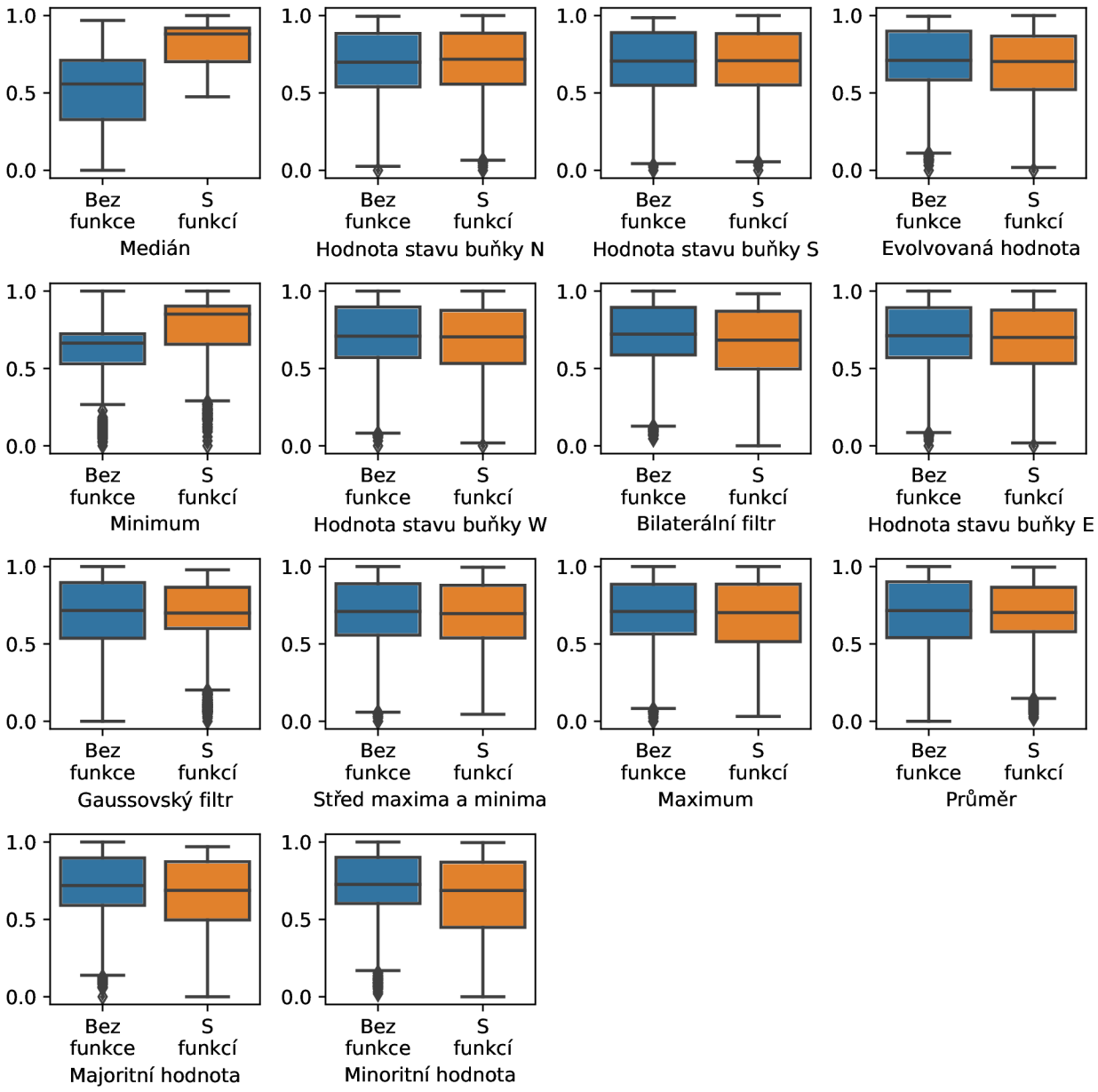
Tabulka 5.2: 10 nejlepších experimentů podle mediánu PSNR pro sadu výpočetních funkcí velikosti 4, kde je každý experiment určen podle zahrnuté výpočetní funkce („T“ pro experimenty s danou funkcí, „F“ bez dané funkce).

	medián PSNR	*median*	*mean*	*min*	*max*	*bilateral*	*gauss*	*half*	*major*	*minor*	*N*	*S*	*W*	*E*	*val*
1	22.003	T	F	F	F	F	F	F	F	F	T	T	F	F	T
2	21.971	T	F	F	F	F	F	T	F	F	F	T	F	F	T
3	21.907	T	F	T	F	F	F	F	F	F	F	F	T	F	T
4	21.897	T	F	F	F	F	F	F	F	F	T	F	T	F	T
5	21.881	T	F	F	F	T	F	F	F	F	F	F	T	F	T
6	21.875	T	F	F	F	F	F	F	F	F	T	T	T	F	F
7	21.844	T	T	F	F	F	F	F	F	F	T	F	F	F	T
8	21.844	T	F	F	F	T	T	F	F	F	T	F	F	F	F
9	21.842	T	F	F	F	F	T	F	F	F	T	T	F	F	F
10	21.826	T	F	F	F	T	F	F	F	F	T	T	F	F	F

Zkoumané kombinace experimentů byly složeny z výpočetních funkcí vyjmenovaných v kapitole 4.3. Kombinace s gaussovým a bilaterálním filtrem byly přidány později a v experimentech, které obsahují gaussův nebo bilaterální filtr, se už neuvažuje polovina maxima a minima, protože se ukázala jako nevýhodná.

Experimenty byly pro variantu se sadou o velikosti 4 a 8 zhodnoceny jednotlivě, nicméně pro shrnutí do jednoho přehledu byly výsledné hodnoty PSNR pro obě sady normalizovány (standardizace pomocí rozpětí), viz graf 5.3. Lze pozorovat, že nehledě na velikost sady (a tedy i typ šumu) mají vyšší kvalitu ty filtry, které mohou na pravou stranu CMR dosadit funkci medián. Další nezanedbatelný vliv má funkce minimum, zejména u sady velikosti 8. U ostatních funkcí nelze určit jednoznačnou odpověď, zda je jejich využití skutečně (ne)výhodné.

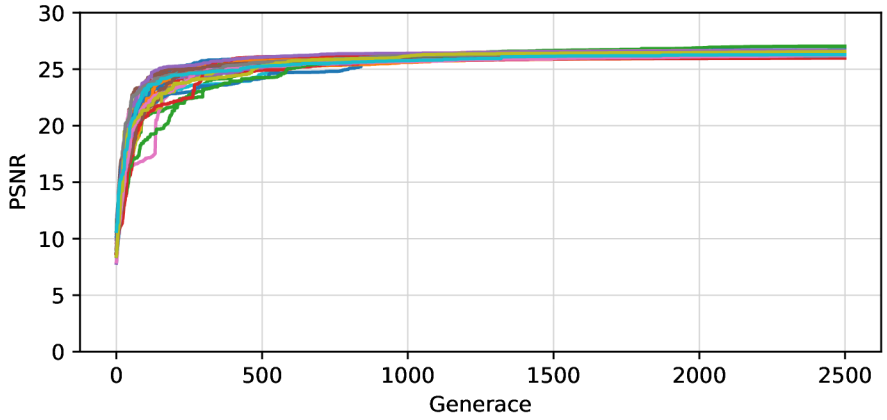
Pro vybrání optimální sady výpočetních funkcí bylo proto zvoleno 10 nejlepších experimentů podle mediánu PSNR pro obě velikosti sad zvlášť, viz tabulky 5.1 a 5.2. Nejvíce využité funkce mezi těmito nejlepšími experimenty pak tvoří vybrané sady, nad kterými jsou v této práci prozkoumávány další experimentální parametry. Vybraná sada s 8 funkcemi se skládá z mediánu, minima, maxima, evolvované konstantní hodnoty a hodnot stavů buněk N, W, E a S. Zvolená sada s 4 funkcemi pak obsahuje medián, evolvovanou konstantní hodnotu a hodnoty stavů buněk N a S. Tyto sady budou pro zjednodušení dále označované jako $rsm8$ a $rsm4$ (ze zkratky „*right side modification*“).



Obrázek 5.3: Statistické vyhodnocení vlivu zařazení různých výpočetních funkcí do podmínkových pravidel. Grafy ukazují normalizovaný medián hodnot PSNR vypočítaných z filtrace trénovacího obrazu. Souhrn je pro variantu se sadou výpočetních funkcí o velikosti 8 nad šůmem sůl a pepř s různou intenzitou na polovinách obrazu 0 % a 60 %, počtem kroků CA 3 a počtem CMR 10, i pro variantu sady 4 výpočetních funkcí nad náhodným šumem o intenzitě 50 %, počtem kroků CA 5 a počtem CMR 20. Normalizace byla provedena standardizací pomocí rozpětí zvlášť pro obě sady.

5.3 Experimenty s vybranými sadami výpočetních funkcí pravé strany CMR

Náplní experimentů provedených v této části práce bylo nalezení co nejkvalitnějších filtrů pro každý z modelových šumů, a zároveň i takový filtr, který by dával vyhovující výsledky u všech modelových šumů. Pro každý modelový šum byla vytvořena porovnání podle jednotlivých parametrů (typ poškození, typ trénovaného šumu, velikost sady funkcí, atd.) a smyslem bylo najít takové parametry, které by nejvíce ovlivňovaly kvalitu získaných filtrů. Připomeňme, že vybranými sadami výpočetních funkcí jsou sada velikosti 8 – $rsm8$ (medián, minimum, maximum, evolvovaná konstantní hodnota a hodnoty stavů buněk N, W, E a S), a sada velikosti 4 – $rsm4$ (medián, evolvovaná konstantní hodnota a hodnoty stavů buněk N a S).



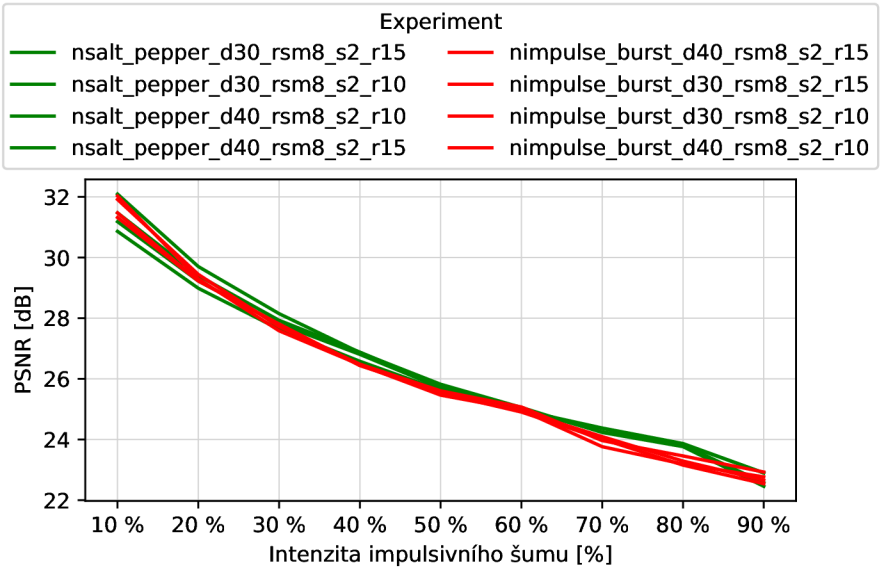
Obrázek 5.4: Konvergenční křivky pro 210 běhů evoluce experimentů s trénovacím obrazem poškozeným šumem sůl a pepř o intenzitě 50 %, 3 kroků CA a všech možných počtů CMR (5, 10, 15, 20, 30, 40, 50). Každý běh zaznamenává nejlepší dosaženou hodnotu PSNR v generaci.

Jak již bylo zmíněno, pro každý experiment bylo provedeno 30 běhů. Při experimentování se ukázalo, že evoluce dosahuje svých nejkvalitnějších filtrů už po přibližně 500 generacích (viz graf 5.4). Téměř všechny běhy experimentu navíc dosahují velmi podobných hodnot nejlepší dosažené hodnoty PSNR. Pro získání dostatečně kvalitního filtru tak postačuje malý počet generací i běhů. Veškeré dosažené filtry v této práci byly navrženy pomocí ES s maximálním počtem generací 2500.

Nejlepší získané filtry pro konkrétní šumy

Jednou z pozorovaných skutečností je schopnost filtru trénovaném na šumu typu sůl a pepř odstraňovat z obrazu i šum *impulse burst*. Tato vlastnost se projevila i u některých referenčních metod (IDBMF, metoda z publikace [40]). Ukázka této vlastnosti je v grafu 5.5, kde jsou porovnány průběhy 4 experimentů trénovaných na šumu sůl a pepř s ekvivalentními experimenty trénovanými na šumu *impulse burst*.

Další ze zkoumaných vlastností nastavení experimentů byl vliv volby jedné ze sad výpočetních funkcí, $rsm8$ nebo $rsm4$, viz grafy 5.6. Zatímco pro šumy typu sůl a pepř a *impulse burst* se jeví jako příznivější sada velikosti 8, u náhodného šumu nehraje velikost sady pří-



Obrázek 5.5: Porovnání experimentů s nastaveními, která se liší pouze typem trénovaného šumu. Experimenty trénované na šumu sůl a pepř jsou vyznačeny zeleně, experimenty trénované na šumu *impulse burst* jsou vyznačeny červeně.

liš velkou roli, což je výhodné. Filtrace náhodného šumu je totiž složitější a nejkvalitnější dosažené filtry tohoto šumu se skládají z většího počtu CMR (40, 50). Menší sada možných výpočetních funkcí tak v tomto případě výrazně snižuje prohledávaný prostor. Naopak u šumů sůl a pepř a *impulse burst* postačoval i nižší počet CMR (10, 15, 20), tudíž nebylo zavedení větší sady problém.

Trénování na konkrétním typu šumu se skutečně ukázalo jako zásadní pro získání kvalitního filtru pro daný model šumu, viz graf 5.8. Pro odstranění šumu sůl a pepř a *impulse burst* se vyplatilo trénovat na šumu sůl a pepř (včetně varianty s kombinací šumů) a trénování pouze na náhodném šumu bylo naopak nevýhodné. Obdobně pro kvalitní filtraci náhodného šumu je vhodné trénovat na náhodném šumu, ačkoli je nutné podotknout, že některé filtry získané trénováním pouze na šumu sůl a pepř dávají také poměrně kvalitní výsledky.

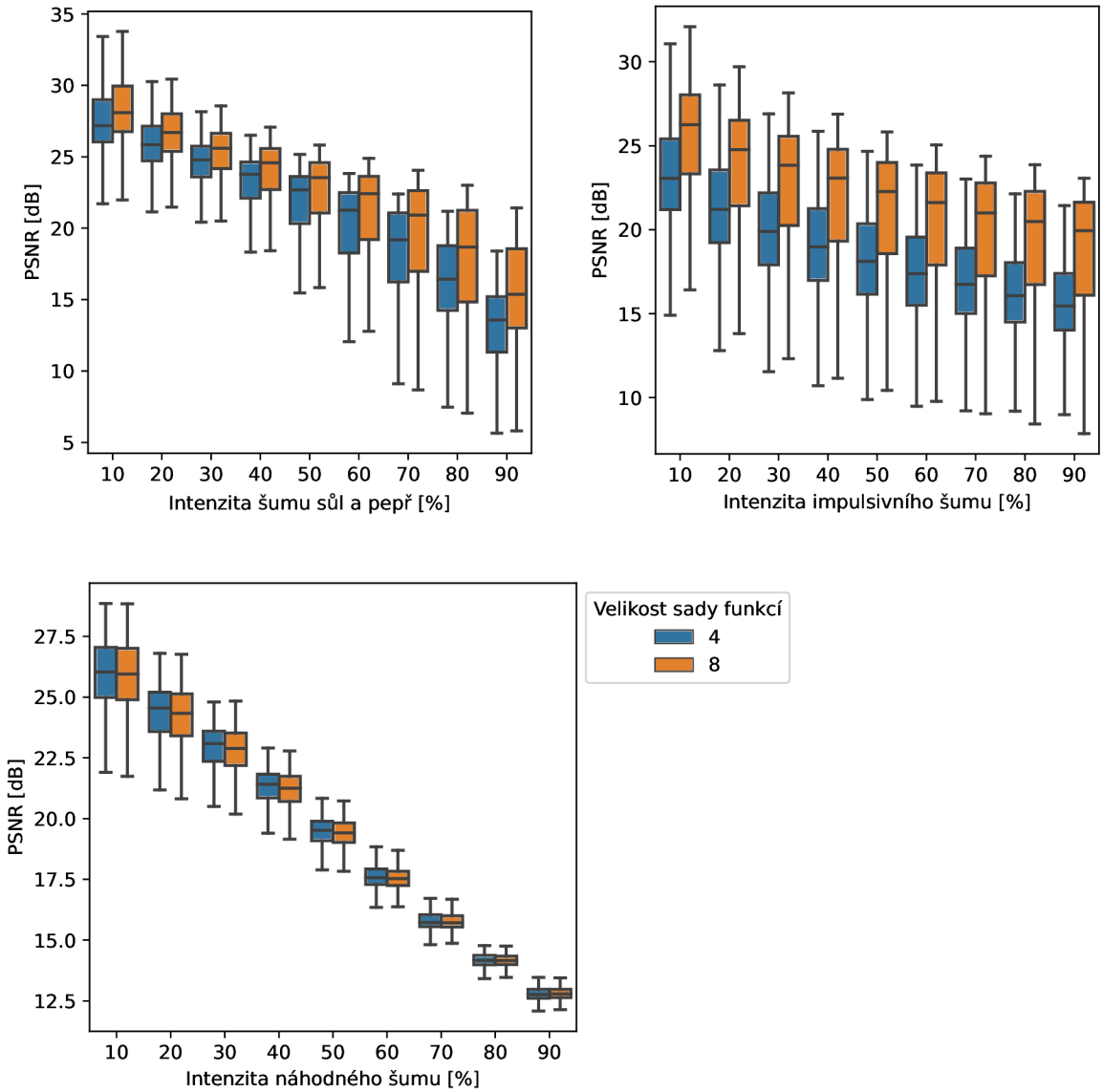
Pro každý modelový šum byly vybrány dva nejlepší experimenty, jeden pro filtraci nižších intenzit šumu (0-50 %) a jeden pro filtraci vyšších intenzit (50-100 %). Z každého experimentu byl vybrán vždy jeden běh a jeho nejlepší dosažený filtr. Pro šum sůl a pepř jsou to filtry $salt1$ z 11. běhu experimentu $nsalt\_pepper\_d30\_rsm8\_s2\_r15$ a $salt2$ z 12. běhu experimentu $nsalt\_pepper\_d60\_rsm8\_s3\_r20$. Pro šum *impulse burst* jsou to filtry $impulse1$ z 18. běhu experimentu $nsalt\_pepper\_d30\_rsm8\_s2\_r15$ a $impulse2$ ze 17. běhu experimentu $nsalt\_pepper\_d30\_rsm8\_s2\_r10$. Pro náhodný šum jsou to filtry $random1$ z 6. běhu experimentu $nrandom\_d30\_rsm8\_s3\_r40$ a $random2$ z 9. běhu experimentu $nrandom\_halved\_d4060\_rsm4\_s5\_r50$.

Nejlepší filtr pro odstranění všech uvažovaných šumů

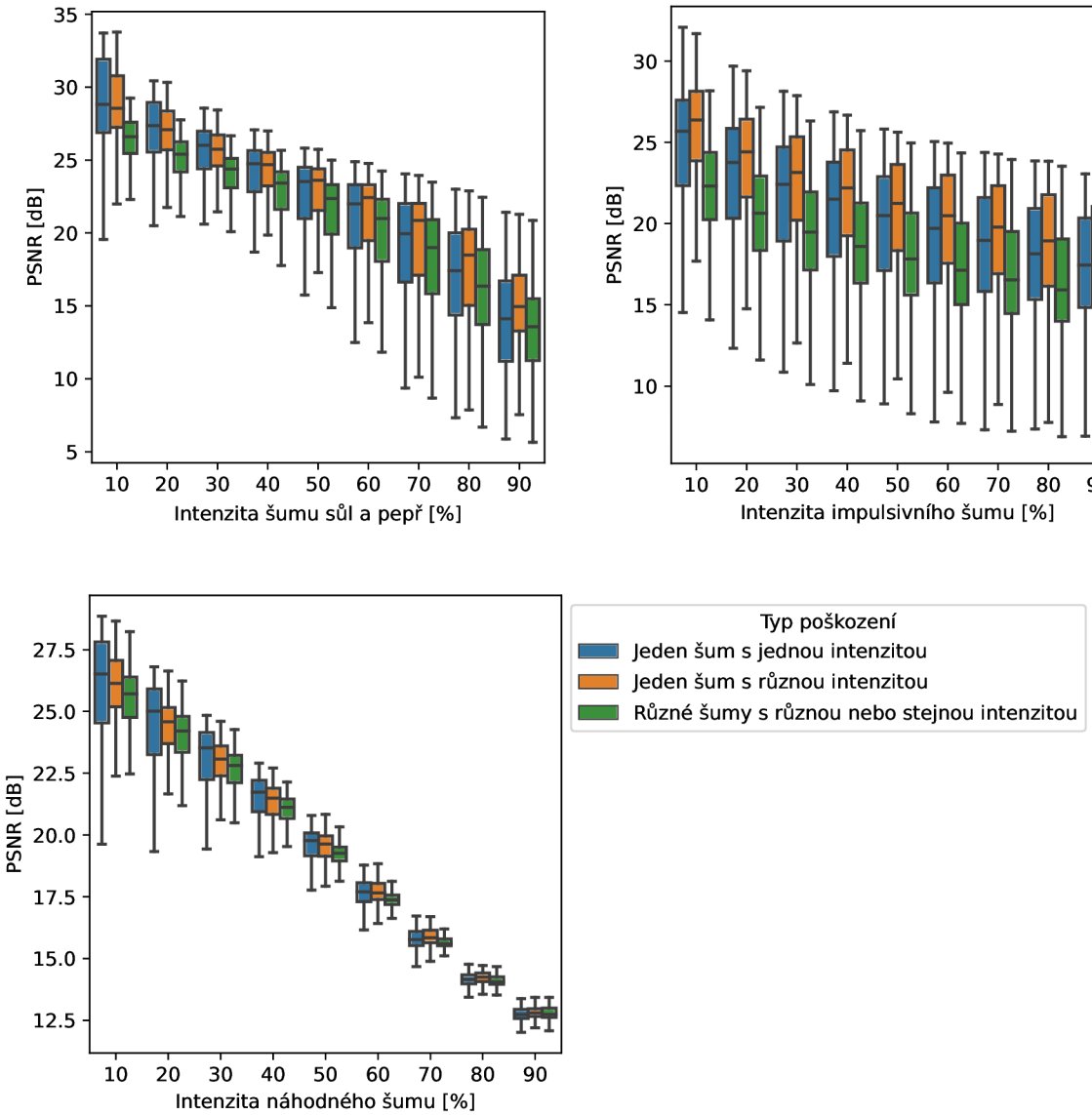
Jednou z velkých ambicí této práce bylo nalézt takový filtr, který by bylo možné uplatnit na filtraci všech uvažovaných modelů šumu. Kvalita filtru pro daný typ šumu se výrazně

odvíjí od zvoleného trénovacího šumu (viz předchozí podkapitola), proto bylo zavedeno trénování na obraze poškozeném různými typy šumu. Takto poškozený obraz měl horní polovinu poškozenou šumem typu sůl a pepř a spodní polovinu náhodným šumem. Zároveň se zužitkovala informace o efektivitě trénování na šumu sůl a pepř při filtrování šumu *impulse burst*. Nejlepší filtry pro konkrétní typ šumu produkovaly experimenty s nastavením poškození pouze jedním typem šumu, přesto kvalita nejlepších filtrů z experimentů trénovaných na obraze poškozeném různými typy šumu příliš nezaostává (viz grafy 5.7). Je však nutné vhodně zvolit i zbylé parametry nastavení experimentu, aby byla filtrace skutečně maximálně kvalitní. Trénování na různých šumech se navíc prokázalo jako klíčové pro určení filtru schopného odstraňovat všechny uvažované modely šumu.

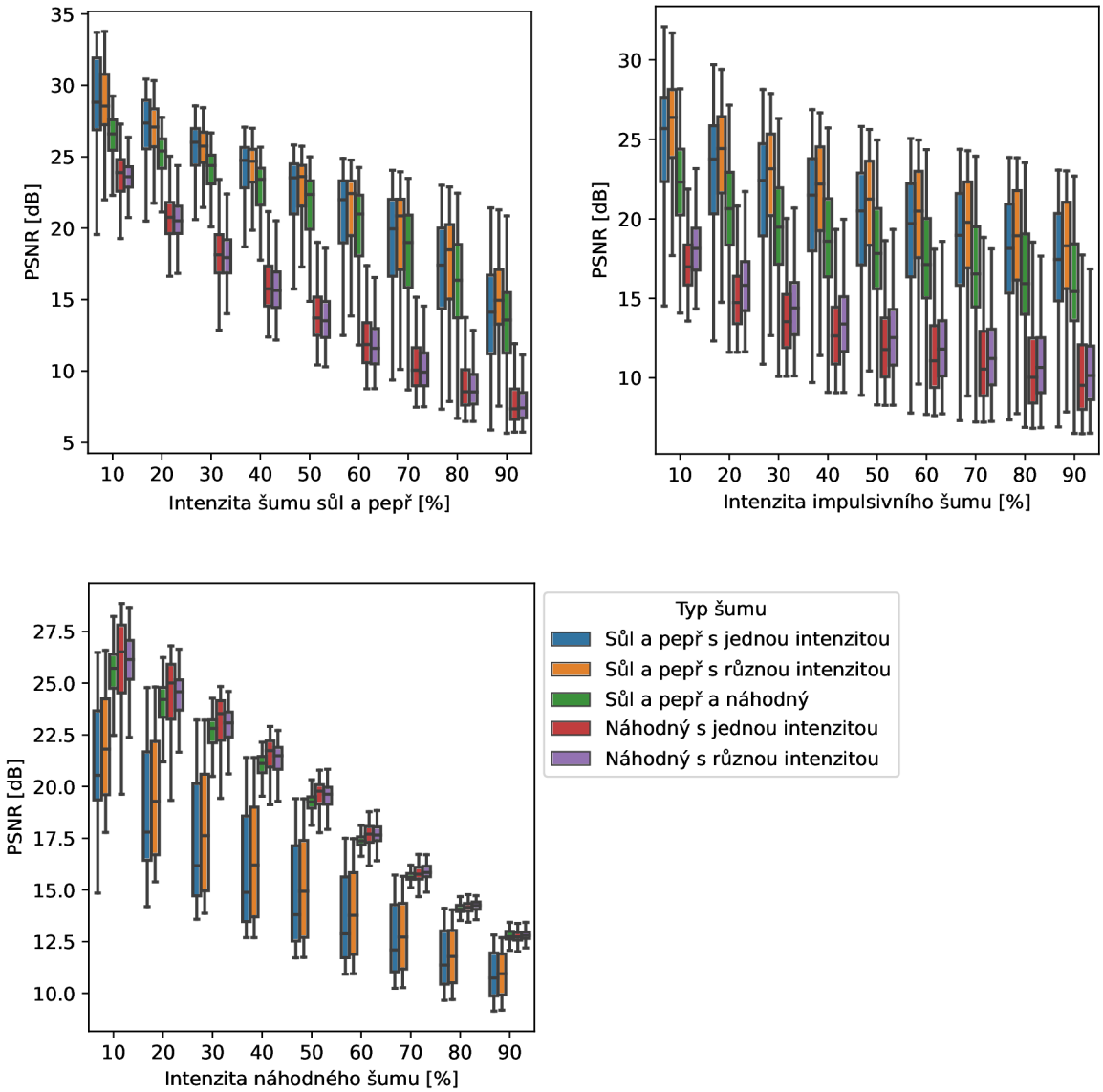
Při určení víceúčelového filtru bylo postupováno následovně. Pro vyhodnocení na každém z modelových šumů byly experimenty seřazeny od nejlepšího k nejhoršímu podle průměrné hodnoty PSNR produkovaných filtrů napříč všemi intenzitami šumu. Poté se hledal takový experiment, který se v pořadníku pro každý modelový šum umístil co nejlépe. Nalezený experiment $nsalt\_pepper\_random\_halved\_d2020\_rsm8\_s1\_r20$ se pro filtraci každého modelového šumu nacházel v první pětině nejlepších experimentů. Konkrétní vybraný filtr experimentu pak pochází z 19. běhu (označení 020) a dále bude pro zjednodušení pojmenovaný jako *multi*.



Obrázek 5.6: Grafy porovnávající experimenty se zvolenou velikostí sady podle jejich průměrné hodnoty PSNR na konkrétních intenzitách modelových šumů.



Obrázek 5.7: Grafy porovnávající experimenty se zvoleným typem poškození podle jejich průměrné hodnoty PSNR na konkrétních intenzitách modelových šumů. Pro šum sůl a pepř a *impulse burst* byly zvoleny jen experimenty trénované na šumu sůl a pepř, pro náhodný šum experimenty trénované na náhodném šumu (modré a oranžové grafy). Všechna porovnání zahrnují experimenty trénované na obraze poškozeném šumy sůl a pepř a náhodným šumem (zelené grafy). Data modrých grafů jsou z experimentů nad jedním typem šumu (šum sůl a pepř, náhodný šum) o jedné intenzitě (10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 %). Oranžové grafy shrnují experimenty také nad jedním šumem (šům sůl a pepř nebo náhodný šum), ale o různých intenzitách (0 % a 10 %, 0 % a 20 %, 0 % a 30 %, 0 % a 40 %, 0 % a 50 %, 0 % a 60 %, 0 % a 70 %, 0 % a 80 %, 0 % a 90 %, 10 % a 60 %, 10 % a 90 %, 20 % a 60 %, 30 % a 60 %, 40 % a 60 %, 40 % a 80 %, 50 % a 70 %). Data zelených grafů jsou z experimentů nad obrazy poškozenými na jedné polovině šumem sůl a pepř a druhé polovině náhodným šumem o různých nebo stejných intenzitách (10 % a 10 %, 20 % a 20 %, 30 % a 30 %, 40 % a 40 %, 50 % a 50 %, 60 % a 60 %, 70 % a 70 %, 80 % a 80 %, 90 % a 90 %, 10 % a 90 %, 90 % a 10 %, 20 % a 80 %, 80 % a 20 %, 40 % a 60 %, 60 % a 40 %).



Obrázek 5.8: Grafy porovnávající experimenty se zvoleným typem trénovaného šumu podle jejich průměrné hodnoty PSNR na konkrétních intenzitách modelových šumů. Data modrých a červených grafů jsou z experimentů nad obrazy poškozenými jednou intenzitou jednoho typu šumu (10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 %). Data oranžových a červených grafů z experimentů nad různými intenzitami téhož šumu (0 % a 10 %, 0 % a 20 %, 0 % a 30 %, 0 % a 40 %, 0 % a 50 %, 0 % a 60 %, 0 % a 70 %, 0 % a 80 %, 0 % a 90 %, 10 % a 60 %, 10 % a 90 %, 20 % a 60 %, 30 % a 60 %, 40 % a 60 %, 40 % a 80 %, 50 % a 70 %). Nakonec data zelených grafů byla získána z výsledků experimentů nad obrazy poškozenými na jedné polovině šumem sůl a pepř a druhé polovině náhodným šumem o různých nebo stejných intenzitách (10 % a 10 %, 20 % a 20 %, 30 % a 30 %, 40 % a 40 %, 50 % a 50 %, 60 % a 60 %, 70 % a 70 %, 80 % a 80 %, 90 % a 90 %, 10 % a 90 %, 90 % a 10 %, 20 % a 80 %, 80 % a 20 %, 40 % a 60 %, 60 % a 40 %).

Výpisy vybraných nejlepších přechodových funkcí CA

Všechny vybrané nejkvalitnější filtry odpovídají přechodové funkci CA obsahující CMR s modifikovanou pravou stranou tak, jak je popsána v kapitole 4.3. V této sekci jsou uvedeny jednotlivé tabulkové reprezentace pro filtry *salt*1, *salt*2, *impulse*1, *impulse*2, *random*1, *random*2 a *multi*. U filtrů pro šum sůl a pepř a *impulse burst* (tabulky 5.3, 5.4, 5.6 a 5.7) lze pozorovat významné zastoupení mediánu a hodnot stavů sousedních buněk na pravé straně CMR. U víceúčelového filtru *multi* (viz tabulka 5.9) je tomu velmi podobně. Zvýšený výskyt konstantní evolvované hodnoty je naopak u filtrů pro náhodný šum (5.5 a 5.8).

Tabulka 5.3: Výpis přechodové funkce CA *salt*1

$c_{i,j-1}$ $(t-1)$	$c_{i-1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i+1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i,j}$ (t)
$= 0$	≥ 23	$= 0$	≥ 99	≤ 38	W
$\neq 0$	≤ 251	≤ 7	$= 0$	≥ 93	N
≤ 112	≥ 146	$= 0$	$\neq 0$	$\neq 0$	*median*
$\neq 0$	≤ 253	≥ 250	≤ 242	≤ 253	*median*
$\neq 0$	$\neq 0$	$= 0$	≥ 44	≥ 18	*median*
$= 0$	≤ 220	≥ 236	$\neq 0$	$\neq 0$	*median*
$= 0$	$\neq 0$	$= 0$	$= 0$	$\neq 0$	W
≤ 216	$\neq 0$	≥ 240	$= 0$	≤ 19	N
≤ 206	≥ 21	≥ 255	≤ 119	≤ 198	S
$\neq 0$	≤ 250	$= 0$	$\neq 0$	$\neq 0$	*median*
≥ 42	≥ 21	$= 0$	$= 0$	$\neq 0$	S
$= 0$	≤ 87	$= 0$	$\neq 0$	$= 0$	E
$\neq 0$	$\neq 0$	≥ 255	$\neq 0$	$\neq 0$	*min*
$= 0$	≤ 175	$= 0$	$\neq 0$	≥ 21	S
$\neq 0$	$\neq 0$	$= 0$	≤ 255	≤ 77	N

Tabulka 5.4: Výpis přechodové funkce CA *salt*2

$c_{i,j-1}$ $(t-1)$	$c_{i-1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i+1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i,j}$ (t)
≥ 21	$= 0$	$= 0$	≤ 99	$\neq 0$	N
$= 0$	$\neq 0$	$= 0$	≤ 32	$= 0$	W
$= 0$	≤ 103	$= 0$	$\neq 0$	$\neq 0$	S
$\neq 0$	$\neq 0$	$= 0$	≤ 139	$\neq 0$	*median*
≥ 254	$\neq 0$	≥ 242	$\neq 0$	$\neq 0$	*min*
$= 0$	$\neq 0$	$= 0$	$= 0$	$\neq 0$	S
≤ 243	≤ 251	≤ 84	$\neq 0$	$\neq 0$	*median*
≥ 22	≤ 200	≥ 253	$\neq 0$	≤ 202	*median*
$\neq 0$	≤ 16	≤ 20	≥ 158	$\neq 0$	*median*
$= 0$	$\neq 0$	≥ 221	≥ 6	≤ 240	*median*
$\neq 0$	$\neq 0$	≥ 246	$\neq 0$	$\neq 0$	N
≤ 237	≤ 226	$\neq 0$	≥ 89	$\neq 0$	*median*
≤ 204	$\neq 0$	≥ 110	≤ 116	$\neq 0$	*median*
$\neq 0$	$\neq 0$	$= 0$	≥ 6	≤ 247	*median*
$\neq 0$	≥ 0	$= 0$	$\neq 0$	≥ 223	N
$= 0$	$\neq 0$	$= 0$	≥ 98	$= 0$	W
$\neq 0$	≤ 143	$= 0$	$\neq 0$	≤ 203	N
$= 0$	$\neq 0$	$= 0$	$\neq 0$	$\neq 0$	*median*
$\neq 0$	≤ 205	≥ 235	≤ 140	$\neq 0$	*median*
$\neq 0$	$\neq 0$	$= 0$	≤ 184	$= 0$	N

Tabulka 5.5: Výpis přechodové funkce CA *random*1

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
≤ 122	$= 0$	$= 0$	≤ 194	≥ 16	E
≤ 52	≥ 180	≤ 181	$= 0$	≤ 91	N
≤ 175	$\neq 0$	≥ 7	$\neq 0$	$= 0$	N
≤ 64	≥ 207	≤ 246	≤ 20	≤ 183	*min*
$\neq 0$	≤ 188	≤ 146	≥ 53	$\neq 0$	*median*
$\neq 0$	≤ 80	≥ 38	≤ 7	≥ 41	S
$= 0$	≥ 117	≤ 1	≤ 154	$\neq 0$	W
≤ 142	≥ 250	$\neq 0$	≤ 93	$\neq 0$	*min*
$\neq 0$	$= 0$	≤ 29	$= 0$	$\neq 0$	132
≥ 156	≤ 234	≤ 17	≤ 68	≤ 200	*min*
≤ 87	≤ 164	$= 0$	$\neq 0$	≤ 186	E
≥ 108	≤ 167	≤ 189	≤ 66	$= 0$	S
≥ 137	$\neq 0$	≤ 37	$\neq 0$	≤ 58	*min*
$= 0$	≤ 204	≥ 138	$\neq 0$	≥ 15	S
≥ 201	$= 0$	≤ 76	≤ 218	$= 0$	W
≤ 137	$= 0$	≤ 115	≤ 52	≥ 52	N
≤ 18	≤ 157	$= 0$	≥ 201	≥ 182	*max*
$= 0$	≥ 234	$= 0$	$= 0$	≤ 254	221
≤ 149	≥ 236	$\neq 0$	≥ 100	≤ 111	*min*
≥ 31	≥ 138	$\neq 0$	≥ 107	≤ 69	*median*

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
≥ 178	≤ 5	$= 0$	$= 0$	$= 0$	*min*
≤ 136	≤ 182	≥ 192	≤ 112	≥ 104	*min*
$= 0$	≤ 33	$\neq 0$	$\neq 0$	$= 0$	S
$= 0$	≤ 39	≤ 216	≤ 58	≤ 80	E
$= 0$	≤ 84	$\neq 0$	≤ 107	≤ 29	*min*
$\neq 0$	≤ 106	≤ 142	$\neq 0$	≥ 131	*min*
$= 0$	$\neq 0$	$\neq 0$	≥ 22	$\neq 0$	*median*
≥ 102	$\neq 0$	$\neq 0$	$= 0$	≥ 27	W
≤ 159	≤ 247	≥ 126	$\neq 0$	≤ 28	S
≤ 3	$\neq 0$	$\neq 0$	≥ 149	≥ 27	217
≥ 149	≥ 39	≤ 127	≤ 151	≤ 104	S
≤ 105	$\neq 0$	≥ 218	$\neq 0$	≤ 204	N
≤ 171	$\neq 0$	≤ 50	$= 0$	$= 0$	91
≤ 190	≤ 205	≥ 211	$\neq 0$	≥ 178	S
$\neq 0$	≤ 53	≥ 223	≥ 29	≤ 36	S
$\neq 0$	≥ 33	$= 0$	≥ 78	≥ 212	*min*
$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	*median*
≤ 246	≥ 137	≤ 208	$\neq 0$	≥ 9	200
$\neq 0$	$\neq 0$	$= 0$	$\neq 0$	≤ 146	253
$\neq 0$	≤ 147	≥ 44	≤ 160	≥ 70	N

Tabulka 5.6: Výpis přechodové funkce CA *impulse*1

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
≤ 246	≤ 252	≥ 253	≤ 252	$\neq 0$	*median*
≤ 251	$\neq 0$	$= 0$	$\neq 0$	$\neq 0$	*median*
≤ 197	≤ 5	$= 0$	≤ 169	$\neq 0$	S
≥ 46	$\neq 0$	≥ 238	≤ 161	$= 0$	*median*
$\neq 0$	$\neq 0$	≥ 255	$\neq 0$	$\neq 0$	*min*
≥ 19	≤ 255	≥ 244	≤ 57	≥ 27	S
≤ 2	$= 0$	$= 0$	≥ 31	≤ 129	E
≤ 218	$\neq 0$	≥ 254	$\neq 0$	≤ 194	*median*
$\neq 0$	$\neq 0$	$= 0$	≤ 92	$\neq 0$	S
$\neq 0$	$\neq 0$	$= 0$	≥ 97	≥ 18	S
$\neq 0$	≤ 213	$= 0$	≥ 15	$\neq 0$	*median*
$\neq 0$	≤ 58	≥ 234	$\neq 0$	≤ 216	*median*
≤ 173	$\neq 0$	$= 0$	≤ 99	≥ 111	W
$\neq 0$	≥ 0	$= 0$	$\neq 0$	$= 0$	N
≤ 111	$\neq 0$	$= 0$	≥ 23	≤ 243	W

Tabulka 5.7: Výpis přechodové funkce CA *impulse*2

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
≥ 5	$\neq 0$	$= 0$	$\neq 0$	$\neq 0$	*median*
≥ 16	$\neq 0$	$= 0$	$= 0$	$\neq 0$	*median*
$\neq 0$	≤ 125	$= 0$	$\neq 0$	≥ 0	N
≥ 248	≥ 13	≥ 242	$\neq 0$	$\neq 0$	*min*
$\neq 0$	≤ 239	≥ 206	≤ 251	≤ 251	*median*
$= 0$	≥ 12	$= 0$	≤ 224	$= 0$	W
$\neq 0$	≥ 41	$= 0$	≤ 255	≤ 63	N
$\neq 0$	$\neq 0$	≥ 243	$\neq 0$	$\neq 0$	N
≤ 235	≤ 238	≥ 245	$\neq 0$	$\neq 0$	*median*
$= 0$	≥ 10	$= 0$	≤ 255	$\neq 0$	S

Tabulka 5.8: Výpis přechodové funkce CA *random2*

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
$\neq 0$	$\neq 0$	≤ 214	$\neq 0$	$\neq 0$	*median*
≥ 55	$= 0$	≤ 209	$= 0$	$= 0$	S
≤ 53	≥ 32	≥ 10	≤ 136	$\neq 0$	2
≥ 184	$\neq 0$	≥ 127	$= 0$	≤ 144	N
≥ 180	$= 0$	$\neq 0$	≤ 180	≥ 199	24
≤ 154	≥ 177	≥ 149	$\neq 0$	≥ 253	186
$= 0$	≥ 153	$\neq 0$	$\neq 0$	≥ 165	S
$\neq 0$	≤ 125	≥ 252	$\neq 0$	≥ 239	6
≥ 47	$\neq 0$	≥ 245	$\neq 0$	$= 0$	S
$\neq 0$	≥ 8	≥ 45	$= 0$	$= 0$	S
≤ 82	≥ 89	$\neq 0$	≤ 16	≤ 98	201
≤ 95	≥ 227	≤ 245	≤ 146	≥ 79	39
≥ 129	≤ 132	$\neq 0$	$= 0$	$= 0$	N
≤ 213	≤ 180	≥ 128	≤ 211	$\neq 0$	3
≥ 8	≤ 239	≥ 121	$= 0$	$\neq 0$	S
≤ 129	≤ 17	$\neq 0$	$\neq 0$	≤ 7	5
$\neq 0$	≥ 239	$\neq 0$	$\neq 0$	≥ 240	S
≤ 38	≥ 114	≤ 53	$\neq 0$	$= 0$	S
≥ 227	$= 0$	≥ 160	≥ 230	$= 0$	246
≤ 106	$= 0$	$= 0$	≤ 17	≥ 63	N
≤ 208	$\neq 0$	≥ 155	≥ 176	$= 0$	*median*
≤ 249	≤ 64	≥ 191	≤ 121	≥ 7	S
≤ 59	$\neq 0$	$\neq 0$	≥ 65	≤ 93	N
≥ 60	≥ 3	≥ 181	≤ 119	$= 0$	60
≤ 223	≥ 89	≥ 201	$\neq 0$	≥ 76	*median*

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
$\neq 0$	≥ 59	$\neq 0$	≤ 203	≥ 119	N
$\neq 0$	≥ 156	≥ 132	≤ 124	$= 0$	S
$\neq 0$	$= 0$	≤ 22	≤ 219	≥ 254	175
$\neq 0$	≥ 192	≥ 104	≥ 190	$\neq 0$	*median*
≤ 71	≤ 57	≤ 216	≤ 163	≤ 183	N
≥ 168	≥ 23	≥ 239	≤ 143	≤ 69	S
$\neq 0$	$\neq 0$	≤ 74	$= 0$	≥ 111	*median*
≤ 103	$\neq 0$	≥ 96	≤ 18	≤ 61	233
≥ 163	$= 0$	≤ 210	≤ 251	$\neq 0$	N
≤ 158	$\neq 0$	≤ 122	≤ 250	≤ 81	206
≥ 172	≤ 201	≤ 106	≥ 0	≤ 37	S
≤ 200	≥ 167	$\neq 0$	≤ 237	≤ 139	2
≤ 31	$= 0$	$= 0$	$\neq 0$	≥ 190	146
$= 0$	≥ 113	≥ 3	≥ 156	≤ 177	N
≤ 243	≥ 186	$\neq 0$	≥ 209	≤ 181	S
≥ 249	$= 0$	$\neq 0$	$= 0$	≤ 236	68
≥ 118	≥ 23	≤ 197	≥ 67	≤ 17	N
$= 0$	$= 0$	≥ 141	≤ 172	$\neq 0$	S
$= 0$	≤ 104	$\neq 0$	≥ 107	$\neq 0$	11
$\neq 0$	$= 0$	≥ 75	≤ 37	≤ 130	*median*
≤ 104	≤ 183	≥ 172	≥ 236	$\neq 0$	N
≤ 188	≤ 29	≤ 182	≤ 58	≥ 60	N
≤ 241	$\neq 0$	≥ 37	≤ 173	≤ 213	229
≥ 75	$\neq 0$	≥ 250	≥ 55	≤ 133	158
≥ 197	$= 0$	≥ 244	≥ 96	≥ 43	N

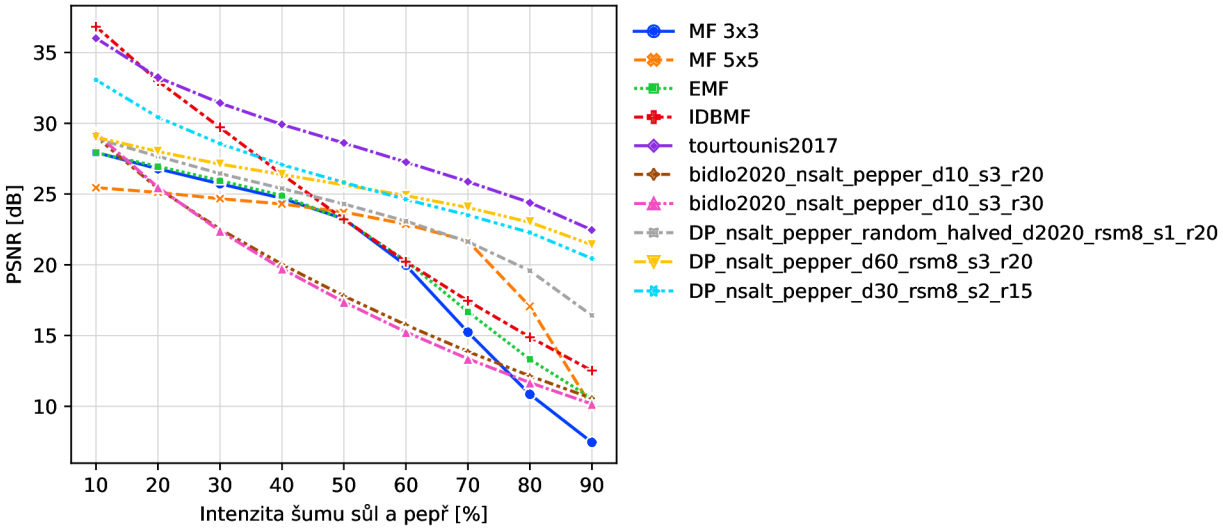
Tabulka 5.9: Výpis přechodové funkce CA *multi*

$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
≥ 196	≥ 32	≥ 211	≤ 27	≥ 16	S
≤ 244	≤ 39	≥ 250	$= 0$	$\neq 0$	S
≤ 101	$\neq 0$	≤ 14	$\neq 0$	$\neq 0$	*median*
$\neq 0$	$= 0$	$= 0$	$= 0$	$\neq 0$	S
$= 0$	$\neq 0$	$= 0$	≤ 237	$\neq 0$	W
$\neq 0$	≤ 254	≥ 174	$\neq 0$	≤ 222	*median*
≥ 29	≤ 206	≤ 11	$= 0$	$= 0$	*max*
≤ 195	$= 0$	$= 0$	$\neq 0$	$= 0$	E
$= 0$	$= 0$	$= 0$	$\neq 0$	$\neq 0$	S
$\neq 0$	≥ 245	≥ 204	$\neq 0$	$= 0$	N

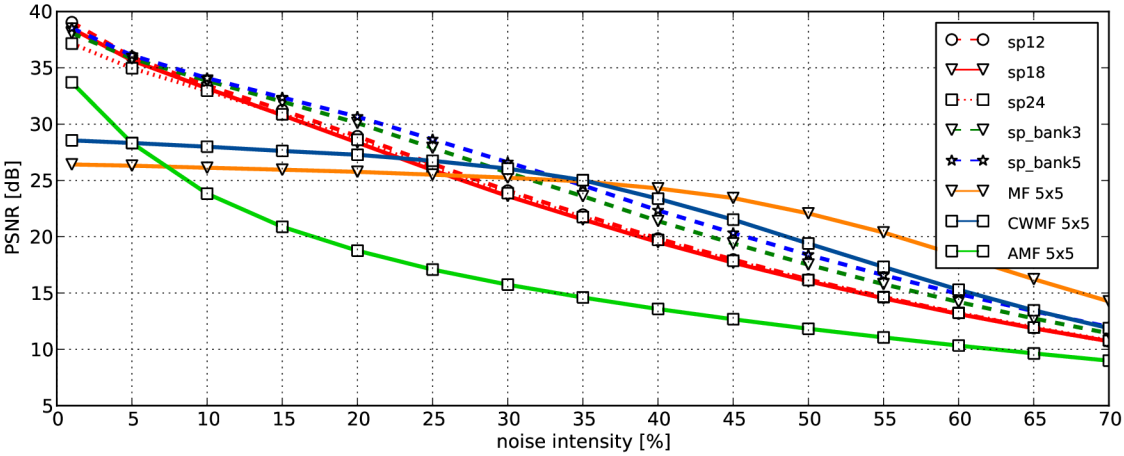
$c_{i,j-1}$ $(t-1)$	$c_{i\ -1,j}$ $(t-1)$	$c_{i,j}$ $(t-1)$	$c_{i\ +1,j}$ $(t-1)$	$c_{i,j+1}$ $(t-1)$	$c_{i\ ,j}$ (t)
≥ 15	$\neq 0$	$= 0$	≤ 52	$= 0$	N
$\neq 0$	$\neq 0$	≥ 218	≤ 171	≤ 58	*median*
≤ 133	≤ 245	$= 0$	≤ 197	$= 0$	W
$\neq 0$	$\neq 0$	$= 0$	≤ 225	$\neq 0$	*median*
$\neq 0$	$\neq 0$	≥ 240	$\neq 0$	$\neq 0$	*min*
≤ 232	≤ 245	≥ 183	$\neq 0$	$\neq 0$	*median*
$\neq 0$	≤ 234	$= 0$	$\neq 0$	$\neq 0$	*median*
≥ 53	$\neq 0$	$= 0$	$\neq 0$	≤ 73	*median*
≥ 66	≥ 224	≤ 43	≥ 203	$\neq 0$	S
≤ 254	$\neq 0$	$\neq 0$	≤ 249	$\neq 0$	*median*

5.4 Výsledky pro šum typu sůl a pepř

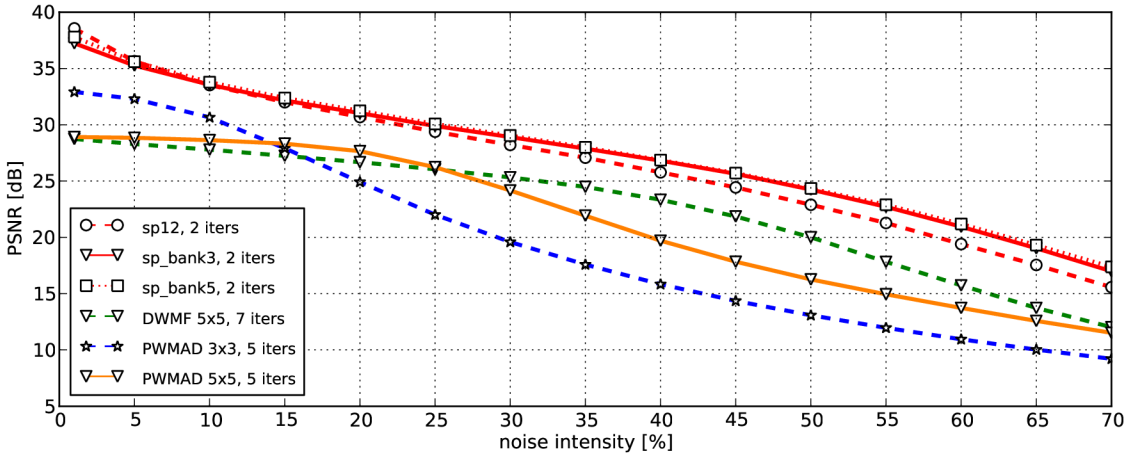
Zvolené nejlepší filtry pro šum sůl a pepř *salt*1 a *salt*2 a nejlepší filtr pro odstranění všech uvažovaných šumu *multi* jsou v této sekci demonstrovány na různých obrazech s různým poškozením. Experimenty, ze kterých tyto filtry vzešly, jsou dále porovnány s ostatními metodami podle průměrné hodnoty PSNR ze všech běhů nad 25 obrazy určenými k vyhodnocení (viz obrázky 5.2).



Obrázek 5.9: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z filtrace sady 25 obrazů (viz 5.2) poškozených šumem sůl a pepř konvenčními metodami (MF 3x3, MF 5x5, IDBMF, EMF), metodou `tourtounis2017`, metodou `bidlo2020` s 20 a 30 CMR, navrhovaným víceúčelovým filtrem a nejlepšími získanými filtry trénovanými na šumu sůl a pepř.



Obrázek 5.10: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z jednokrokové filtrace sady 30 obrazů poškozených šumem sůl a pepř metodou `vasicek2013` (označení *sp*). Převzato z [43]



Obrázek 5.11: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z vícekrokové filtrace sady 30 obrazů poškozených šumem sůl a pepř metodou `vasicek2013` (označení *sp*). Převzato z [43]

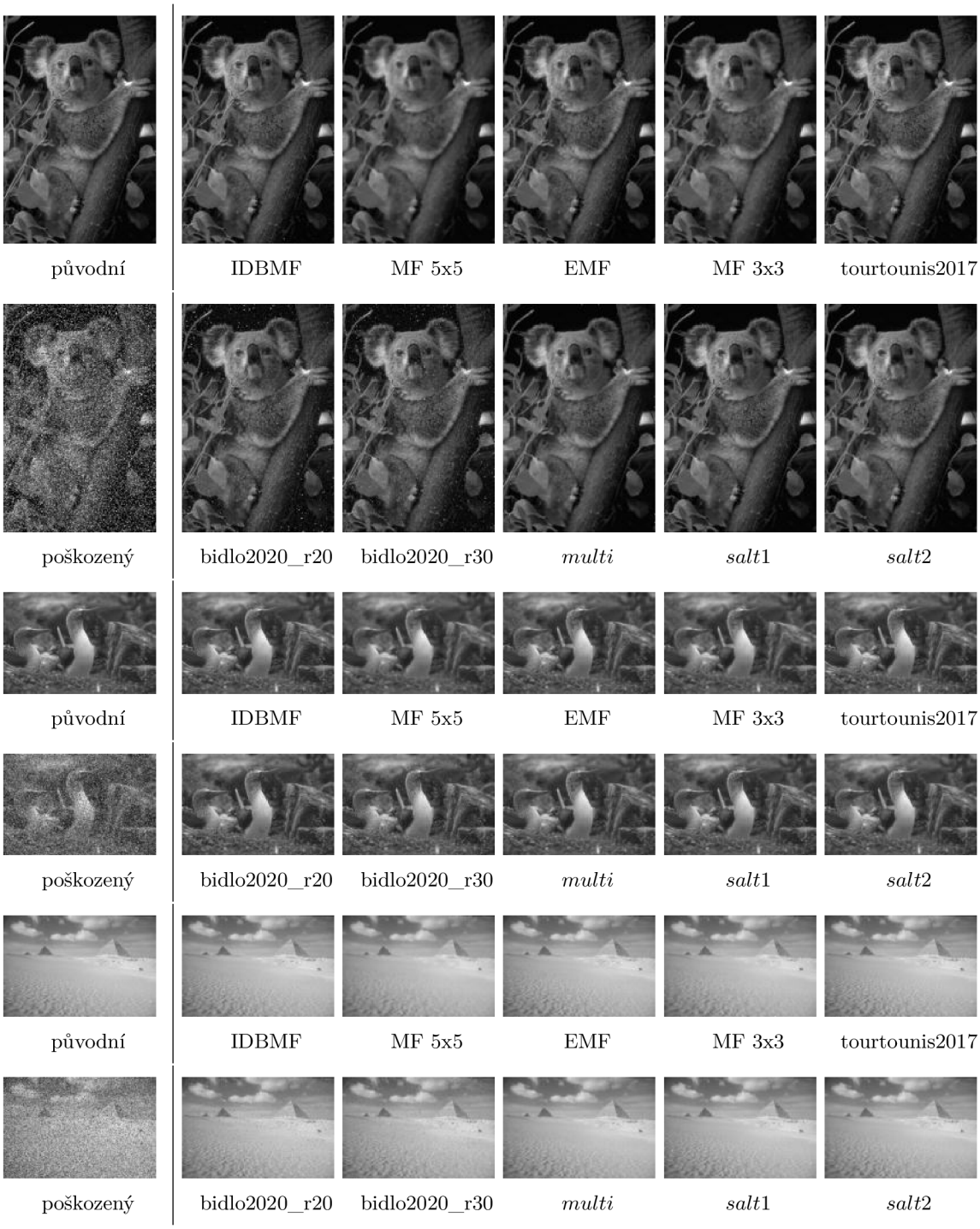
Navrhovaná metoda s vybranými nastaveními experimentů (v grafu 5.9 jako *nsalt_...*) dává bezesporu lepší výsledky oproti původní uvažované metodě [8] (dále jako `bidlo2020`). Zejména při vyšších intenzitách šumu lze pozorovat výrazné zlepšení oproti téměř všem metodám kromě metody [40] (dále jako `tourtounis2017`). U nižších intenzit nepatří navrhované filtry mezi úplně nejlepší, avšak rozdíly ve vizuální kvalitě filtrace jsou malé. Kvalita filtrace metody z publikace [43] (dále jako `vasicek2013`) je shrnuta v grafech 5.10 a 5.11. Oproti této metodě je navrhovaná metoda méně kvalitní na nižších intenzitách šumu a také potřebuje více iterací aplikace filtru (kroků filtrace). Na druhou stranu je nutné upozornit na fakt, že navrhovaná metoda pracuje pouze s von Neumanovým okolím (5-okolím), zatímco `tourtounis2017` s Moorovým okolím (9-okolím) a `vasicek2013` dokonce s filtračním oknem velikosti 5x5. Z porovnání s ukázkami z článku [43] (viz obrázky 5.12) se navrhované nejlepší filtry ukazují jako srovnatelně kvalitní. Metody IDBMF a `tourtounis2017` navíc ve své implementaci přímo pracují s informací, že budou poškozené pixely nabývat hodnot 0 nebo 255.

Následují ukázky filtrace vybranými navrženými filtry vůči porovnávaným metodám. Porovnání s metodou `vasicek2013` je v obrázcích 5.12, porovnání se zbylými metodami na 3 intenzitách je v obrázcích 5.13, 5.14, 5.15 a výřezech 5.16, 5.17 a 5.18. Na výřezech je dobře vidět, že filtr *salt*1, který byl trénován na nižší intenzitě šumu, dobře zachovává detaily a ostrost hran v porovnání s filtry *salt*2 a *multi*. Mediánové filtry MF 3x3 i 5x5 v tomto ohledu selhávají i na nižších intenzitách, byť mají relativně vysoké hodnoty PSNR. U vyšších intenzit šumu už pak dávají srovnatelně kvalitní výsledky pouze navrhované filtry a metoda `tourtounis2017`. Ostatní metody buď ponechávají větší množství poškození (MF 3x3, IDBMF, EMF, `bidlo2020`) nebo v případě MF 5x5 se jedná o silně rozmazané (u 90% šumu nečitelné) obrazy. Víceúčelový filtr *multi* je sice méně kvalitní než filtry *salt*1 a *salt*2, přesto dává srovnatelné kvalitní výsledky na všech intenzitách, byť u velmi vysokých intenzit zanechává větší zbytky poškození.

Ukázka filtrace na obrazech z metody `vasicek2013`

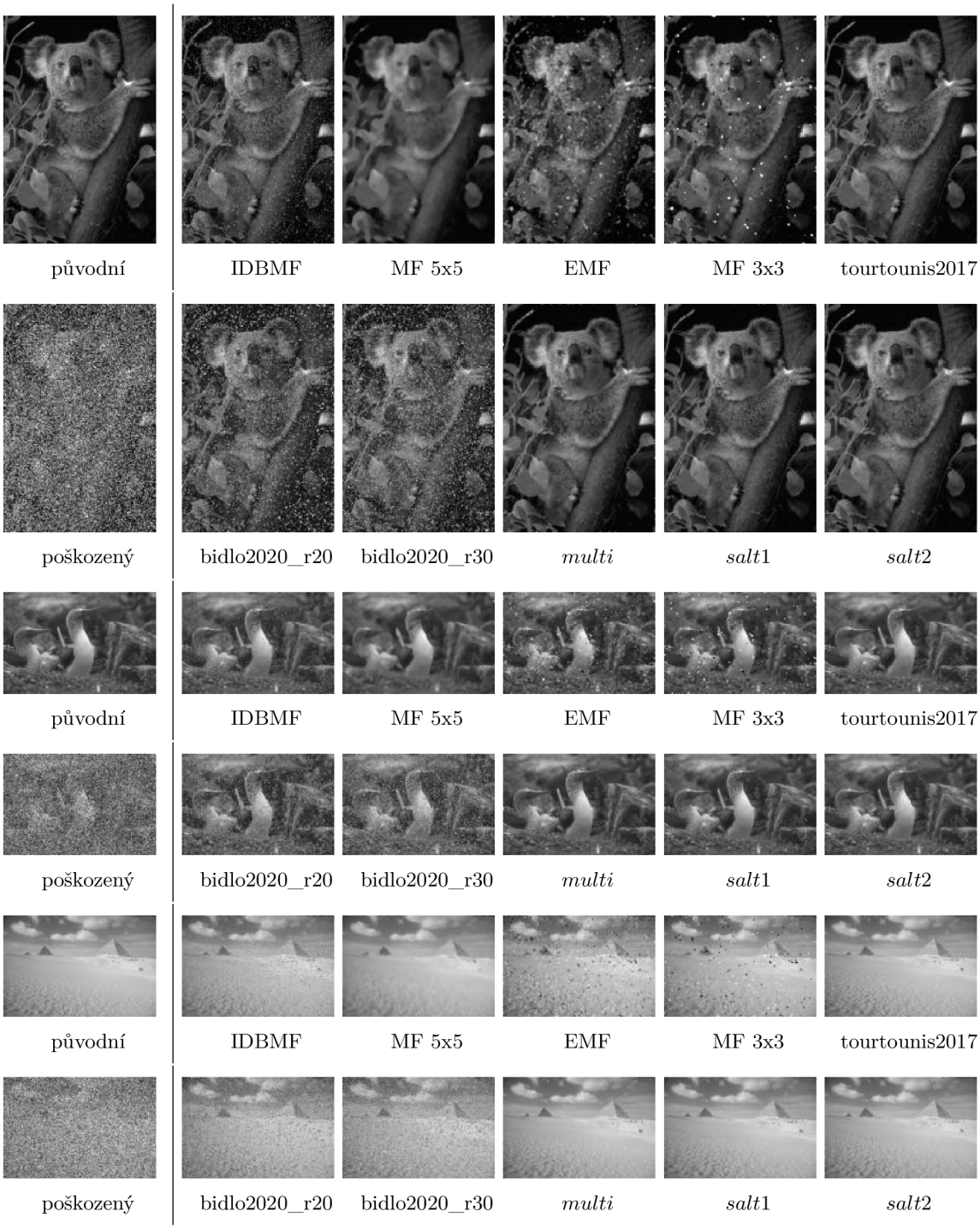
Obrázek 5.12: Porovnání kvality filtrace s metodou `vasicek2013` na obrazech 317080 (srna) a 189011 (chlapec) poškozenými šumem sůl a pepř s intenzitami 5 % a 30 % respektive. Vlevo (označení vasicek2013, převzato z [43]) je výsledek aplikace filtrů metody `vasicek2013`, vpravo je výsledek aplikace navrhovaných filtrů. Filtry metody `vasicek2013` byly vyhodnoceny po 1 nebo 2 krocích (*2 iters*), navrhované filtry jsou vyhodnoceny po 2 krocích pro 5% šum a po 3 krocích pro 30% šum.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 30 %



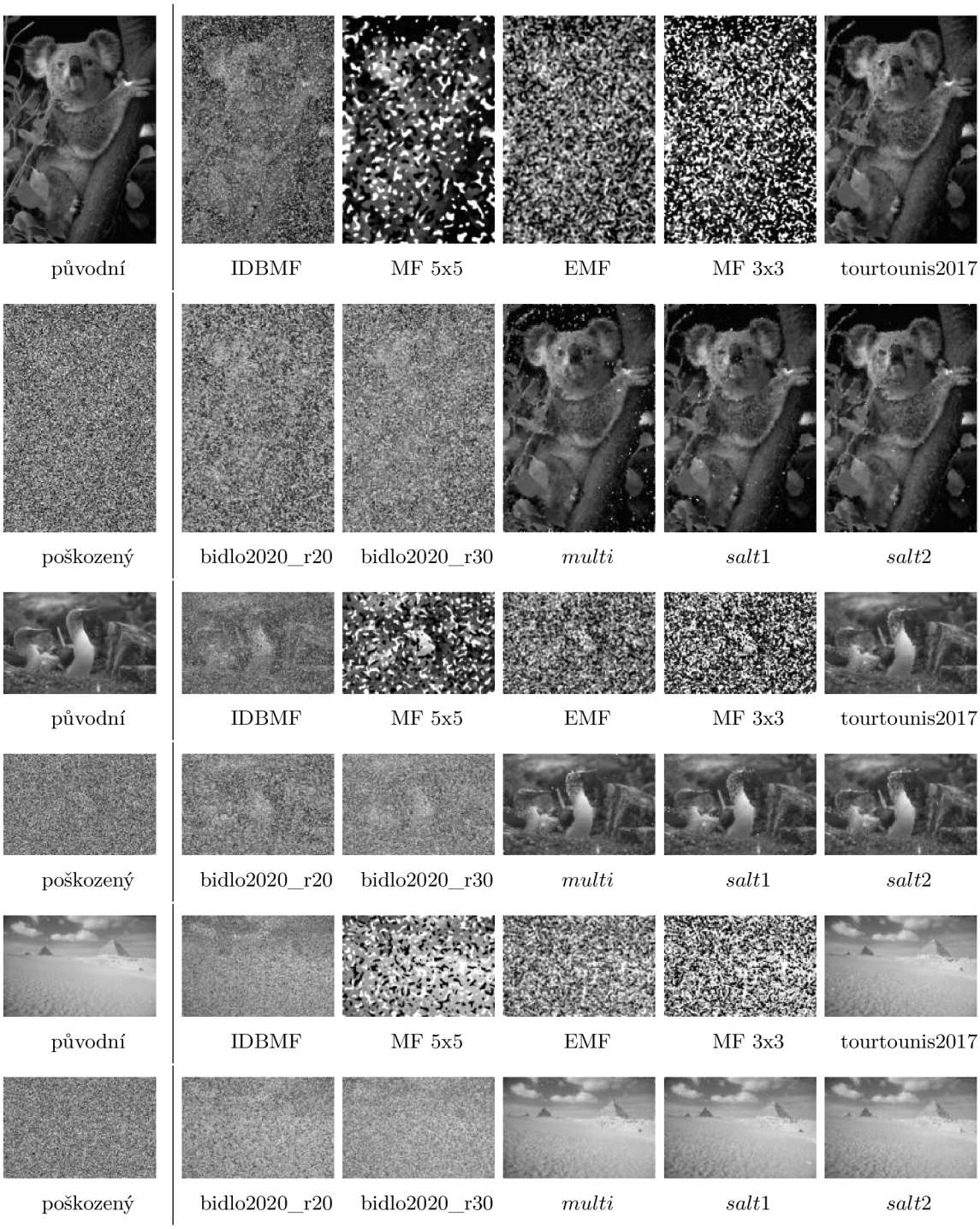
Obrázek 5.13: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené šumem sůl a pepř s intenzitou 30 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR, navrženého víceúčelového filtru *multi* a filtry trénované na šumu sůl a pepř *salt*1 a *salt*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 4 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 60 %



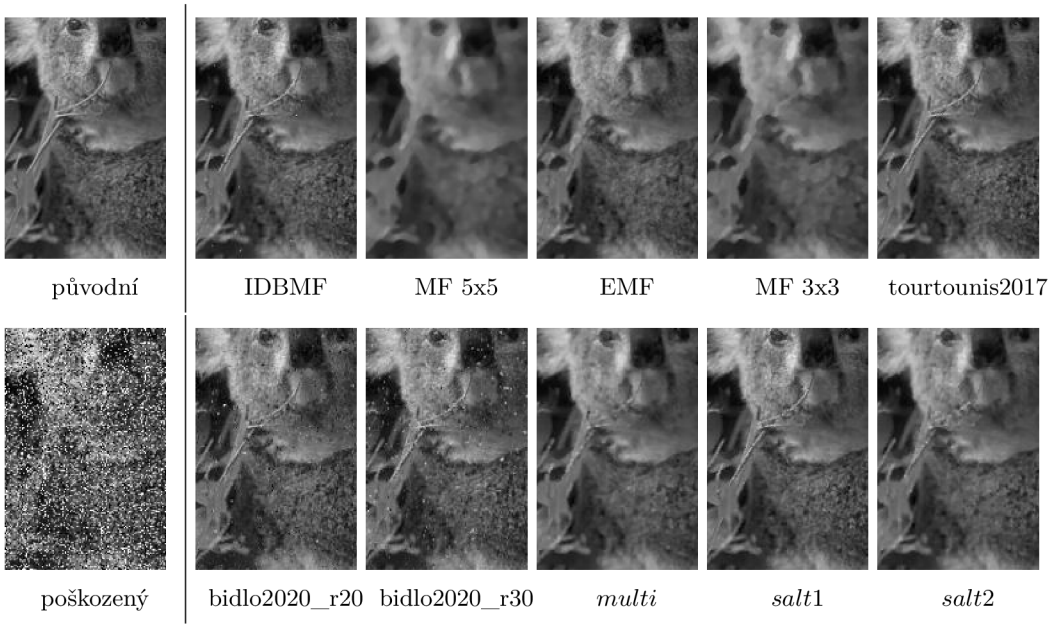
Obrázek 5.14: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené šumem sůl a pepř s intenzitou 60 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR, navrženého víceúčelového filtru *multi* a filtry trénované na šumu sůl a pepř *salt*1 a *salt*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 7 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 90 %

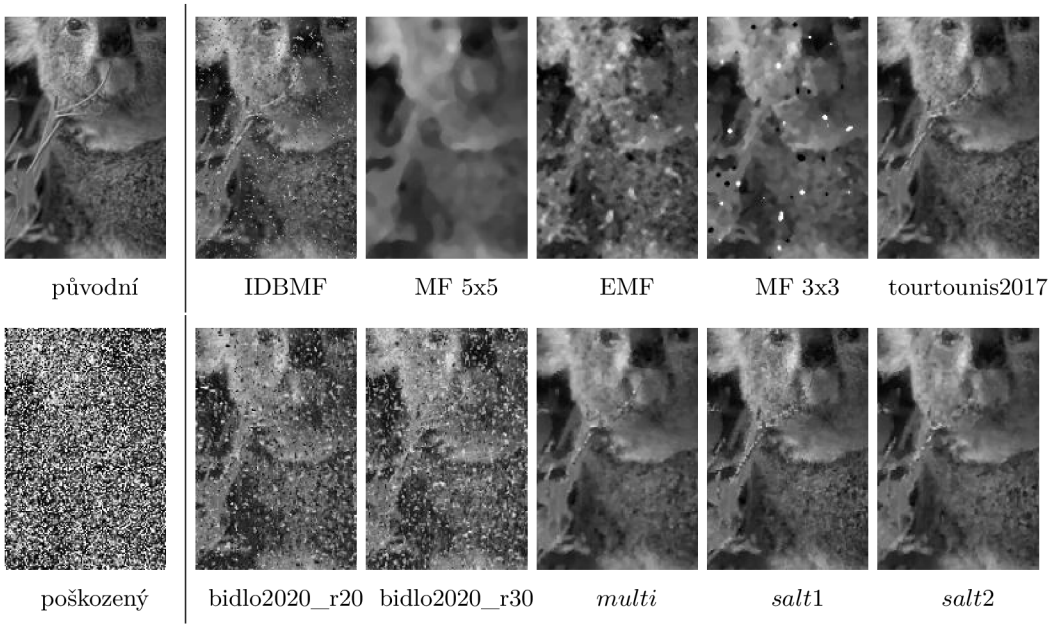


Obrázek 5.15: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené šumem sůl a pepř s intenzitou 90 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR, navrženého víceúčelového filtru *multi* a filtry trénované na šumu sůl a pepř *salt*1 a *salt*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 10 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

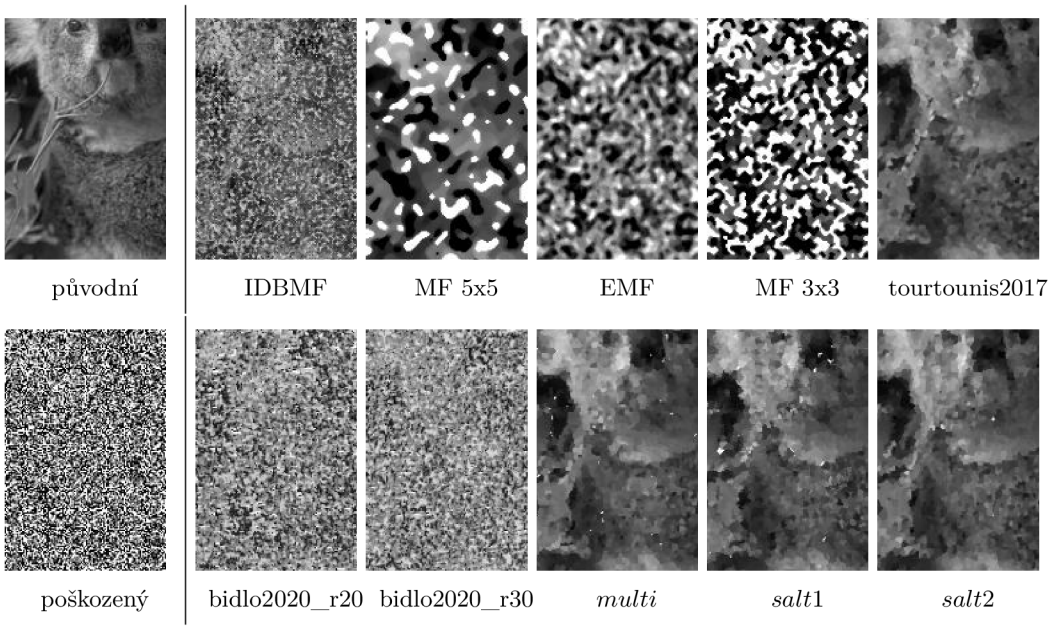
Výřezy z ukázek filtrace



Obrázek 5.16: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.13



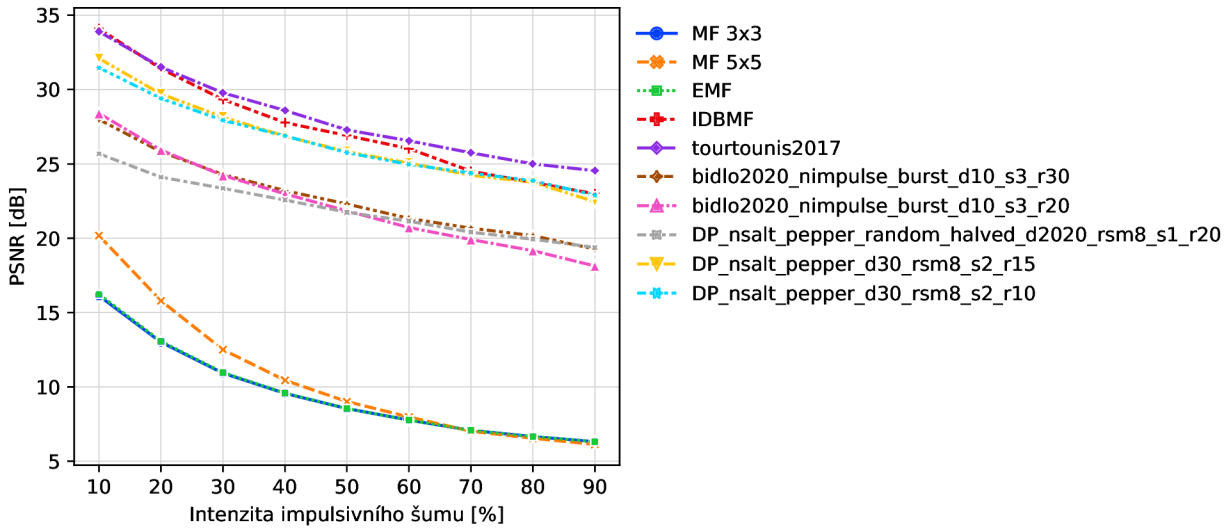
Obrázek 5.17: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.14



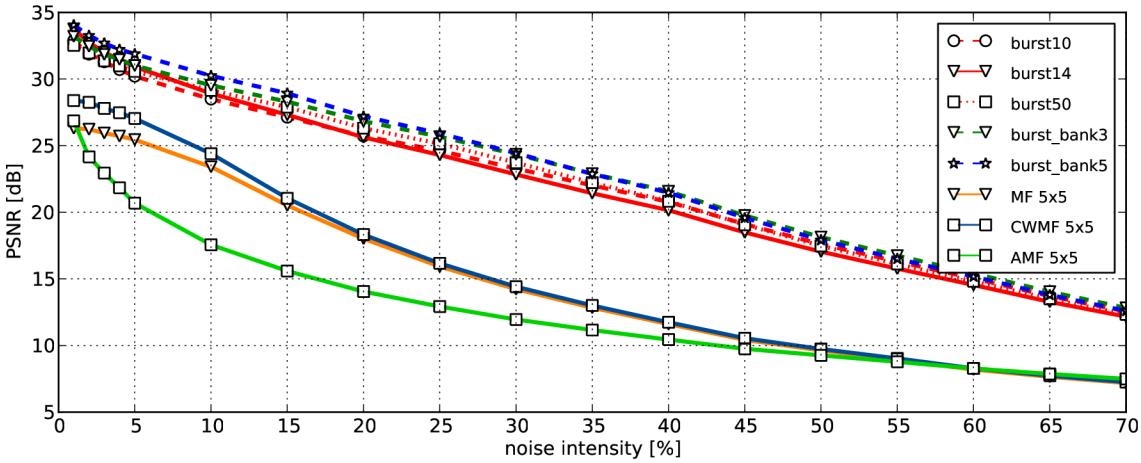
Obrázek 5.18: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.15

5.5 Výsledky pro šum *impulse burst*

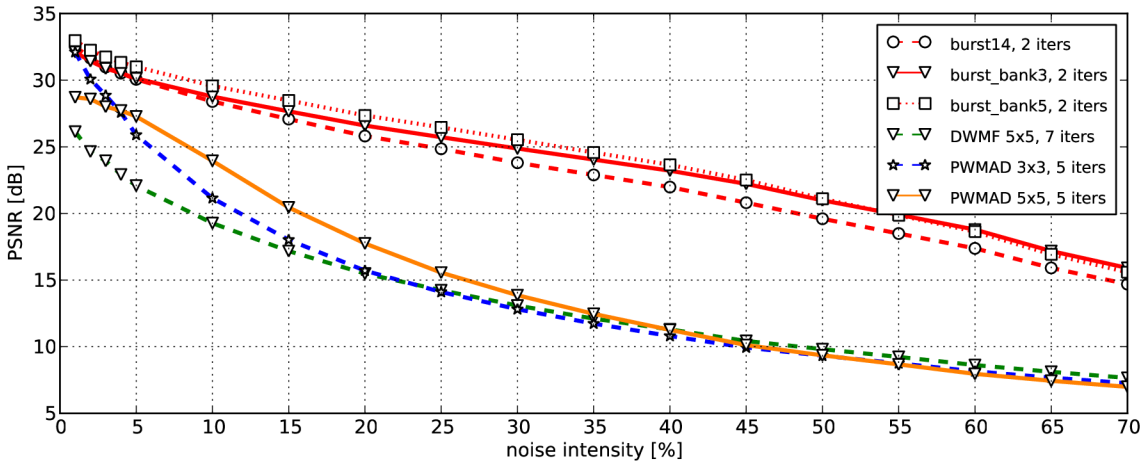
Podobně jako u výsledků pro šum sůl a pepř, i v této sekci jsou rozebrány dosažené výsledky při odstraňování šumu *impulse burst* s vybranými filtry *impulse*1, *impulse*2 a *multi*. Je důležité znovu připomenout, že byla během experimentování zjištěna schopnost filtrů trénovaných na šumu sůl a pepř filtrovat i šum *impulse burst*, a to ve stejné kvalitě, jako kdyby byly trénované na šumu *impulse burst*. Této skutečnosti využívají všechny vybrané filtry.



Obrázek 5.19: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z filtrace sady 25 obrazů (viz 5.2) poškozených šumem *impulse burst* konvenčními metodami (MF 3x3, MF 5x5, IDBMF, EMF), metodou `tourtounis2017`, metodou `bidlo2020` s 20 a 30 CMR trénovanou na šumu *impulse burst*, navrhovaným víceúčelovým filtrem a nejlepšími získanými filtry trénovanými na šumu sůl a pepř.



Obrázek 5.20: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z jednokrokové filtrace sady 30 obrazů poškozených šumem *impulse burst* metodou `vasicek2013` (označení *burst*). Převzato z [43]

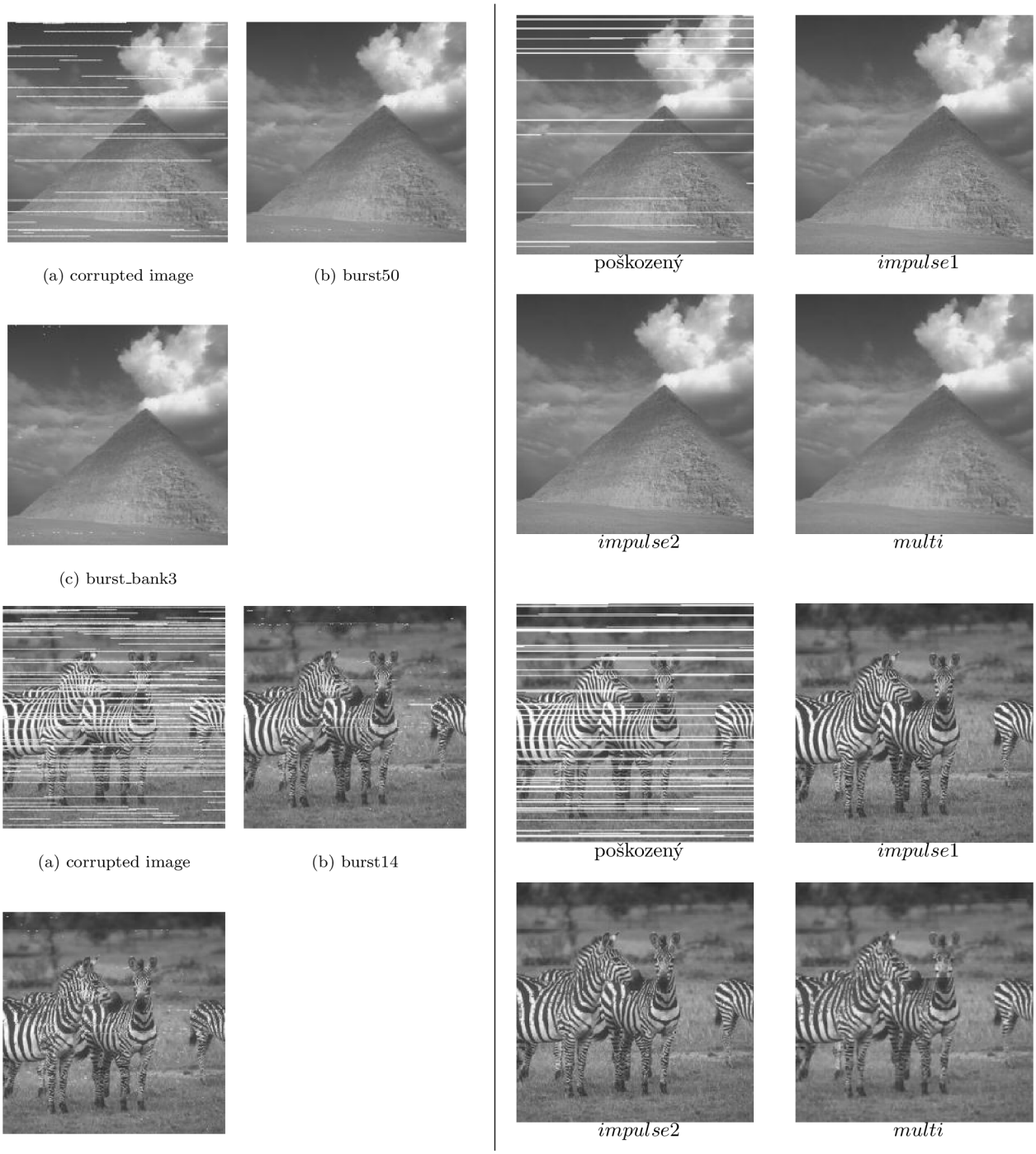


Obrázek 5.21: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z vícekrokové filtrace sady 30 obrazů poškozených šumem *impulse burst* metodou `vasicek2013` (označení *burst*). Převzato z [43]

Téměř žádná z metod kromě metody `vasicek2013` nebyla na tento šum cílena, i přesto byly v rámci porovnání s navrhovanou metodou všechny vybrané metody vyhodnoceny na šumu *impulse burst*. Z grafu 5.19 je vidět, že navrhovaná metoda je schopná produkovat kvalitní filtry šumu *impulse burst* spolu s metodami `tourtounis2017`, IDBMF a `vasicek2013`. Podobně jako u navrhované metody, i u metod `tourtounis2017` a IDBMF se projevila schopnost dobře filtrovat šum *impulse burst*, přestože jsou cílené na šum sůl a pepř. Naopak mediánové filtry MF 3x3 a 5x5 a EMF naprosto selhávají a s tímto typem šumu si neporadí ani při nízkých intenzitách. Metoda `bidlo2020` byla pro účely porovnání přetrénovaná na šumu *impulse burst* (původní nastavení uvažuje pouze šum sůl a pepř) a i díky tomu je schopná kvalitní filtrace tohoto šumu. Víceúčelový filtr *multi* je v tomto ohledu na podobné úrovni jako přetrénovaná metoda `bidlo2020`, což může být považováno za úspěch, neboť na tento šum cílí pouze částečně (trénování na obraze poškozeném na horní polovině šumem sůl a pepř a na dolní polovině náhodným šumem).

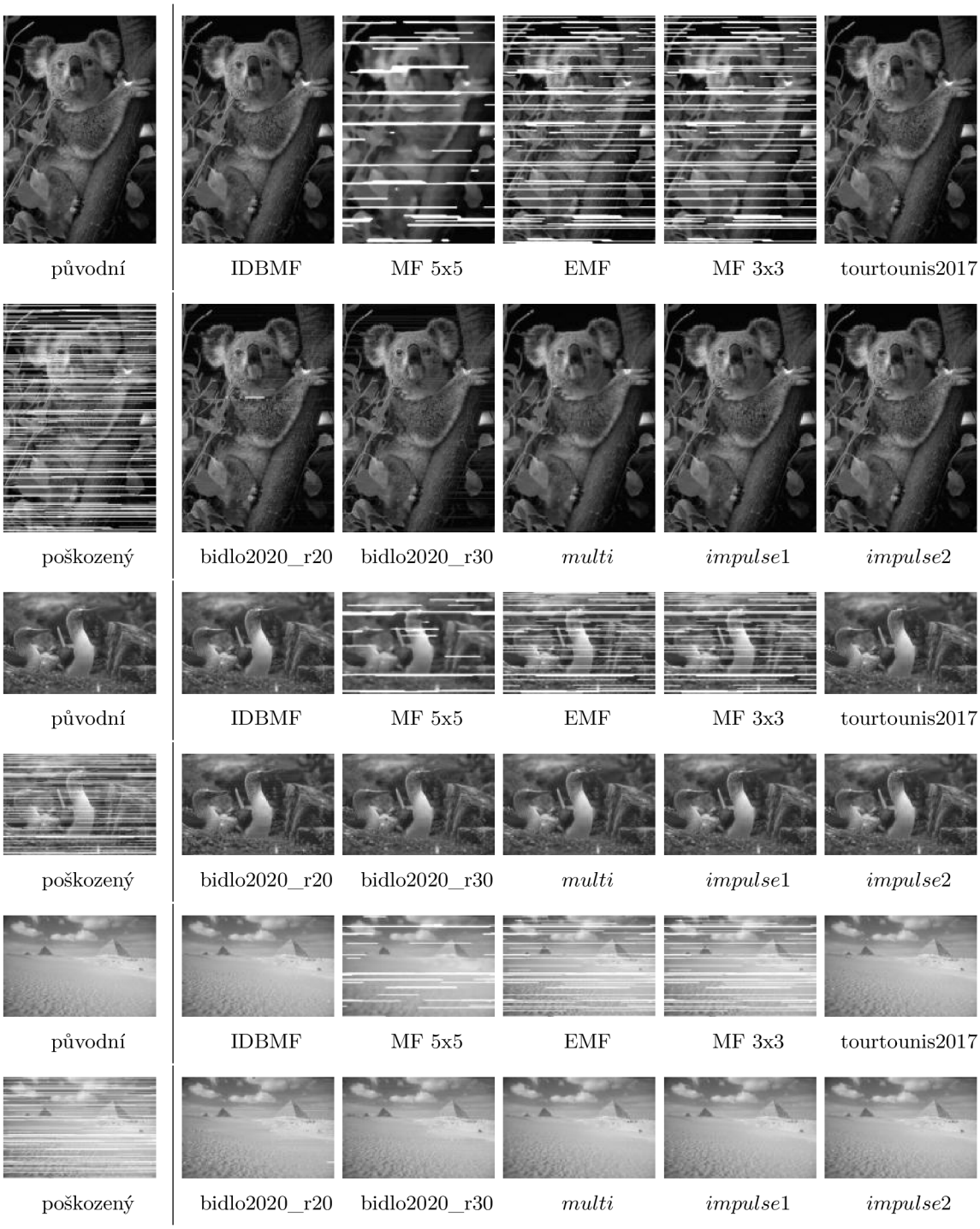
Stejně jako u šumu sůl a pepř byly i pro tento šum vytvořeny ukázky filtrace na různých intenzitách a obrazech. Porovnání s metodou `vasicek2013` je v obrázcích 5.22, s ostatními metodami pak v 5.23, 5.24 a 5.25 a ve výřezech 5.26, 5.27 a 5.28. U nižších intenzit je možné pozorovat dobré zachování detailu a hran u navrhovaného filtru *impulse*1 cíleného na nižší intenzity šumu *impulse burst*. Ostatní navrhované filtry a metody `vasicek2013` a `bidlo2020` zanechávají v obraze pásové fragmenty typické pro šum *impulse burst*. Většina ze zmíněných metod, které si s daným šumem poradí, ho dokáže dobře filtrovat i u vysokých intenzit. Zásadní nevýhoda navrhované metody je v omezení von Neumannovým okolím (vodorovné pásy obvykle poškodí polovinu sousedních pixelů), ačkoli dává lepší výsledky než vícekroková varianta metody `vasicek2013` pracující s filtračním oknem 5x5. Vzhledem k povaze modelového šumu *impulse burst*, kdy poškozené sekvence pixelů nabývají hodnoty 255, se u metod IDBMF a `tourtounis2017` opět vyplatilo vyřazování pixelů s touto hodnotou při výpočtu mediánu (průměru).

Ukázka filtrace na obrazech z metody `vasicek2013`



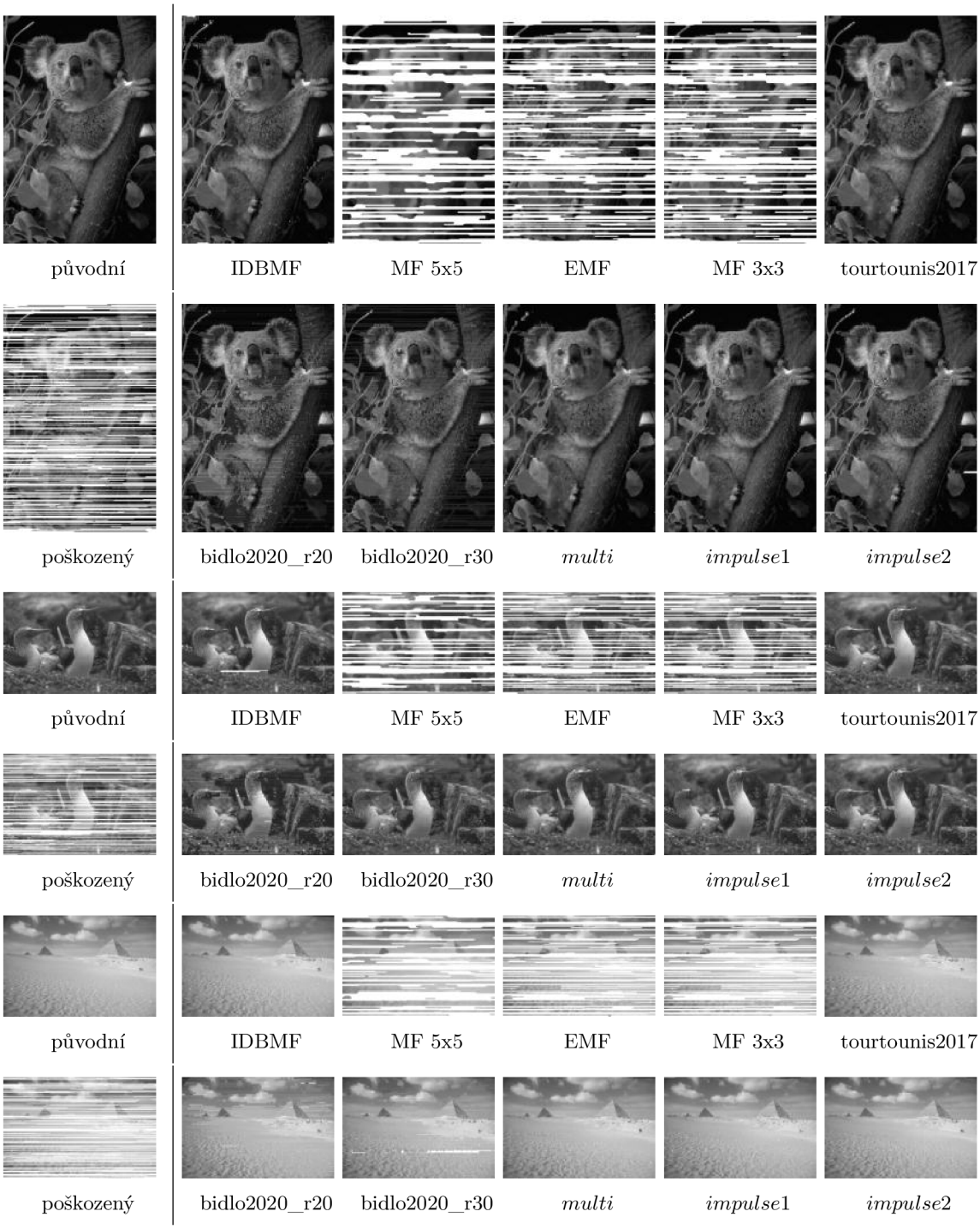
Obrázek 5.22: Porovnání kvality filtrace s metodou `vasicek2013` na obrazech 299091 (pyramida) a 253027 (zebry) poškozenými šumem *impulse burst* s intenzitami 5 % a 20 % respektive. Vlevo (označení vasicek2013, převzato z [43]) je výsledek aplikace filtrů metody `vasicek2013`, vpravo je výsledek aplikace navrhovaných filtrů. Filtry metody `vasicek2013` byly vyhodnoceny po 1 nebo 2 krocích (*2 iters*), navrhované filtry jsou vyhodnoceny po 1-2 krocích pro 5% šum a po 3-4 krocích pro 20% šum.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 30 %



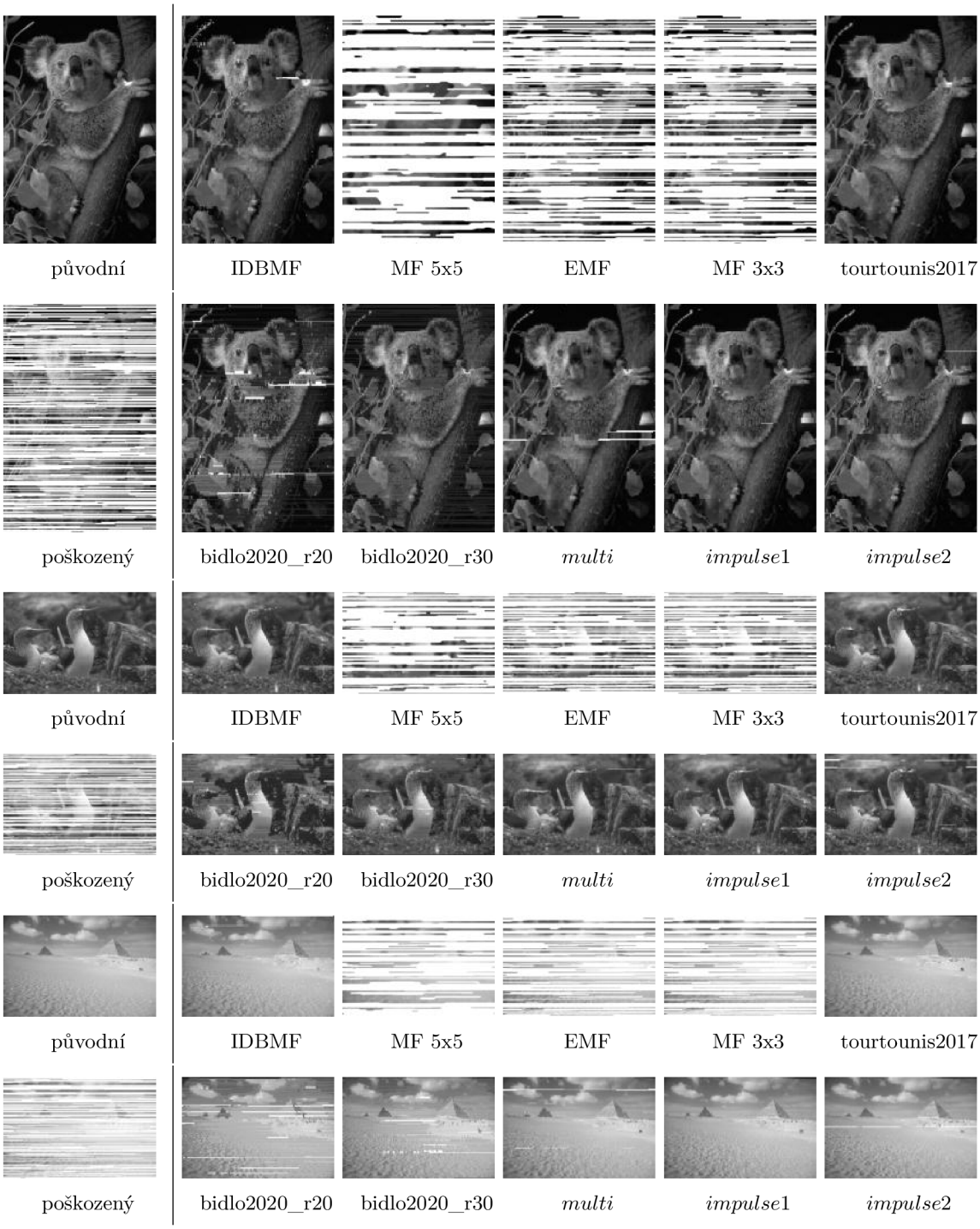
Obrázek 5.23: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené šumem *impulse burst* s intenzitou 30 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR trénované na šumu *impulse burst*, navrženého víceúčelového filtru *multi* a filtry trénované na šumu *impulse burst* *impulse*1 a *impulse*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 4 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 60 %



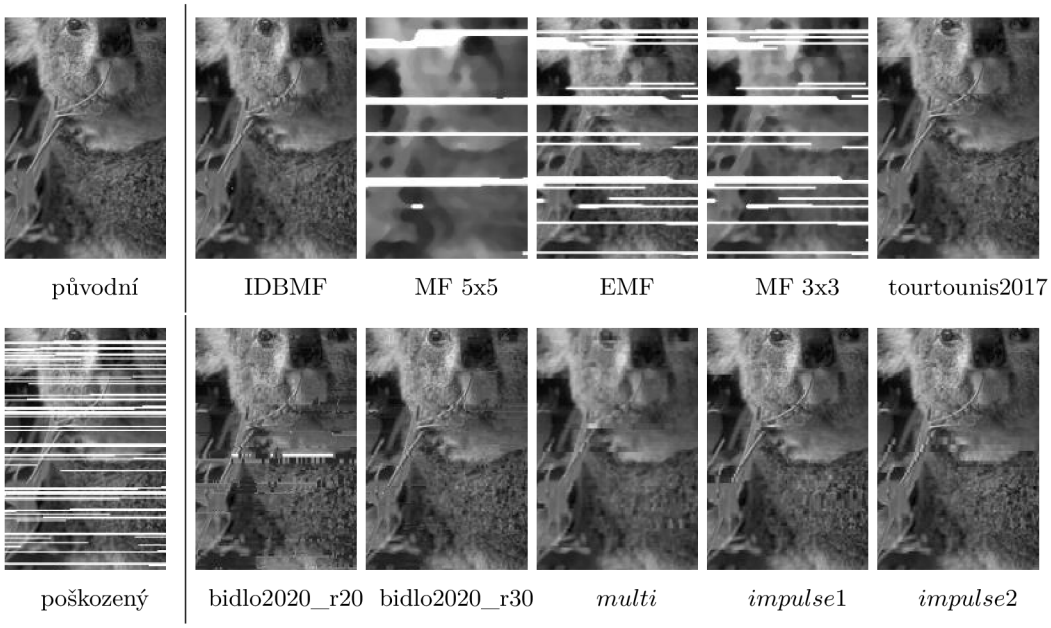
Obrázek 5.24: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené šumem *impulse burst* s intenzitou 60 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR trénované na šumu *impulse burst*, navrženého víceúčelového filtru *multi* a filtry trénované na šumu *impulse burst* *impulse*1 a *impulse*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 7 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 90 %

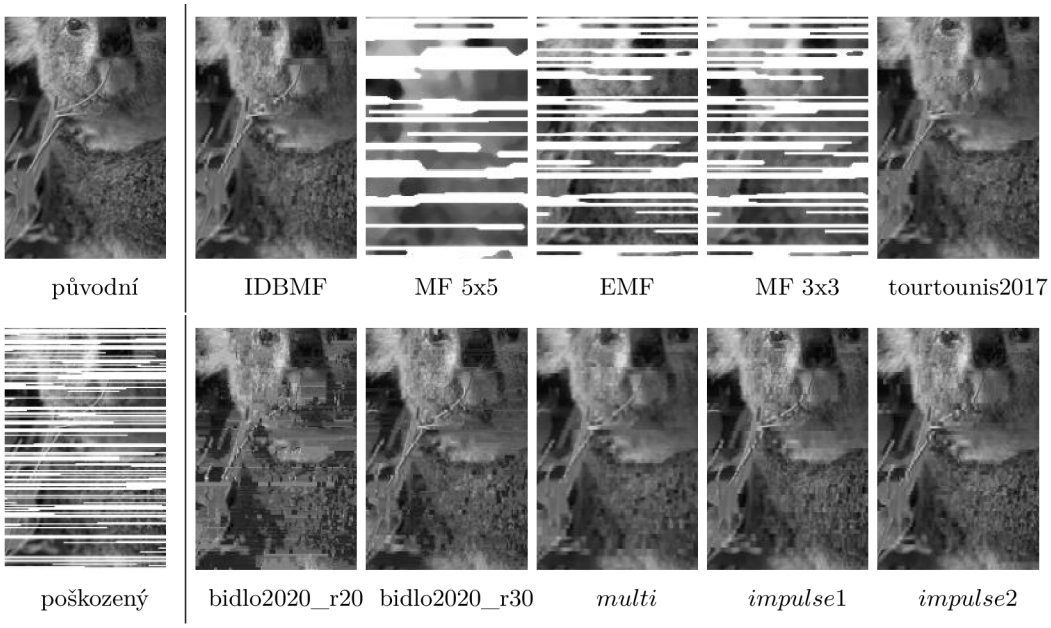


Obrázek 5.25: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené šumem *impulse burst* s intenzitou 90 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR trénované na šumu *impulse burst*, navrženého víceúčelového filtru *multi* a filtry trénované na šumu *impulse burst* *impulse*1 a *impulse*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 10 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

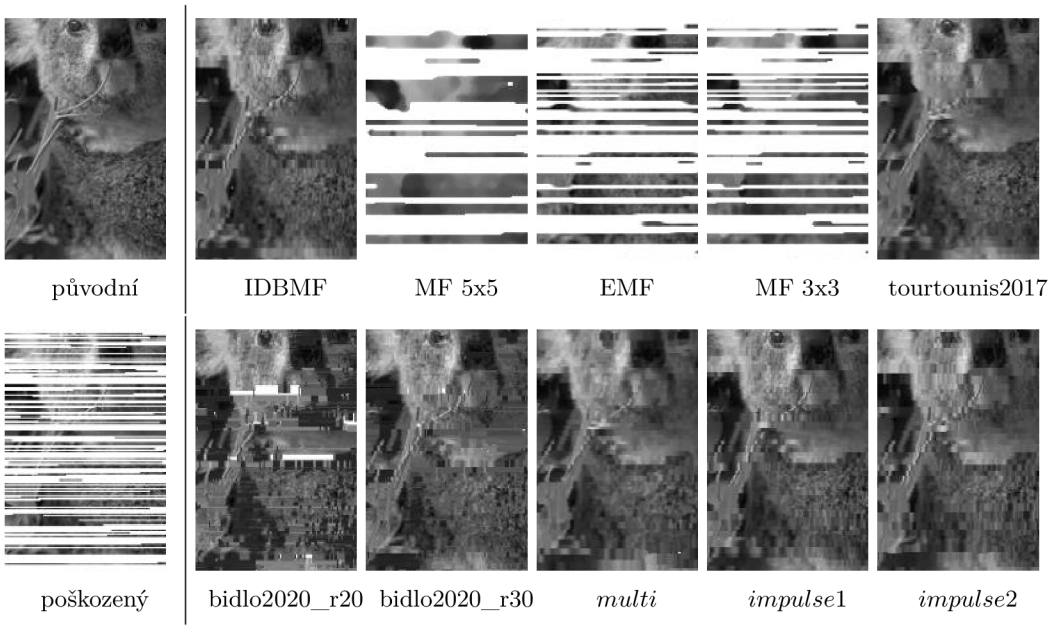
Výřezy z ukázek filtrace



Obrázek 5.26: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.23



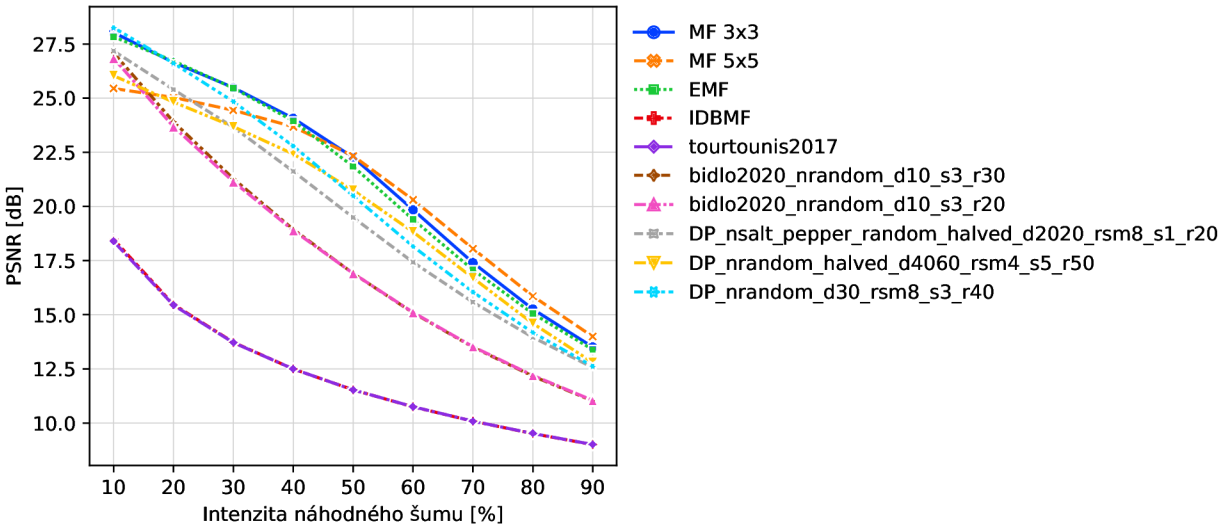
Obrázek 5.27: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.24



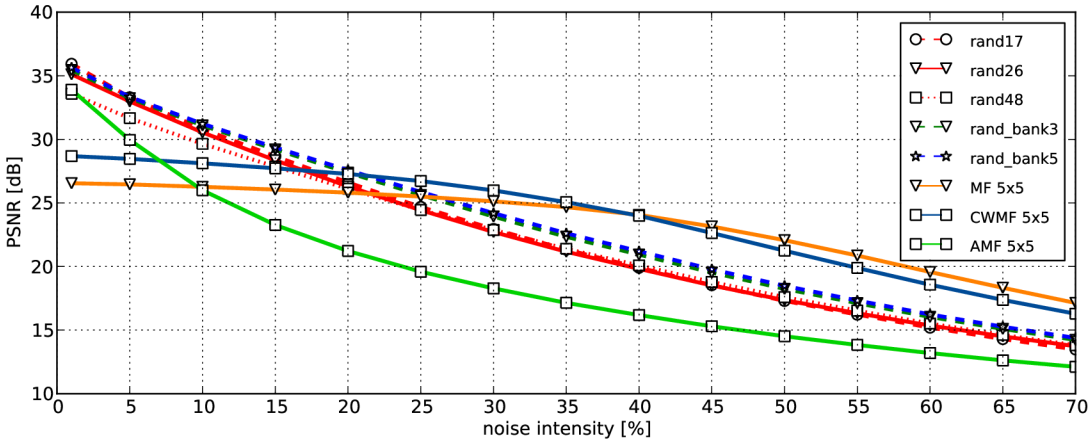
Obrázek 5.28: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.25

5.6 Výsledky pro náhodný šum

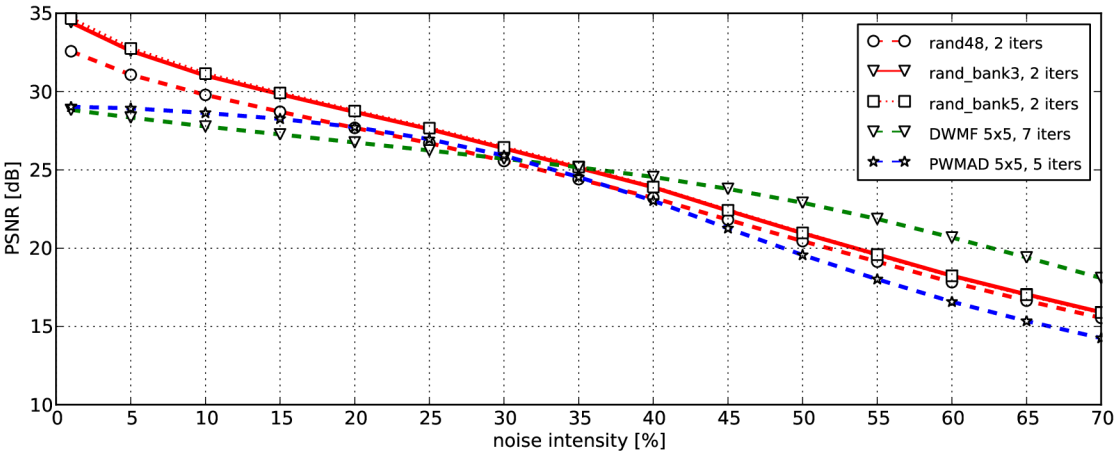
Posledním z modelových šumů, jehož filtrací se tato práce zabývá, je náhodný šum, pro který byly zvoleny filtry *random*1, *random*2 a *multi*. Filtrace náhodného šumu je obtížná, protože na rozdíl od šumů sůl a pepř a *impulse burst* není poškození pixelu omezeno na konkrétní hodnotu. Hodnota poškozeného pixelu se tak může lišit jen málo a detekce je tak náročná. Většina zkoumaných filtrů, včetně navrhované metody, nebyla schopna dostatečně kvalitně filtrovat vyšší intenzity (60 % a více), proto jsou ukázky v této sekci limitované do 50% intenzity šumu.



Obrázek 5.29: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z filtrace sady 25 obrazů (viz 5.2) poškozených náhodným šumem konvenčními metodami (MF 3x3, MF 5x5, IDBMF, EMF), metodou `tourtounis2017`, metodou `bidlo2020` s 20 a 30 CMR trénovanou na náhodném šumu, navrhovaným víceúčelovým filtrem a nejlepšími získanými filtry trénovanými na náhodném šumu.



Obrázek 5.30: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z jednokrokové filtrace sady 30 obrazů poškozených náhodným šumem metodou `vasicek2013` (označení *rand*). Převzato z [43]

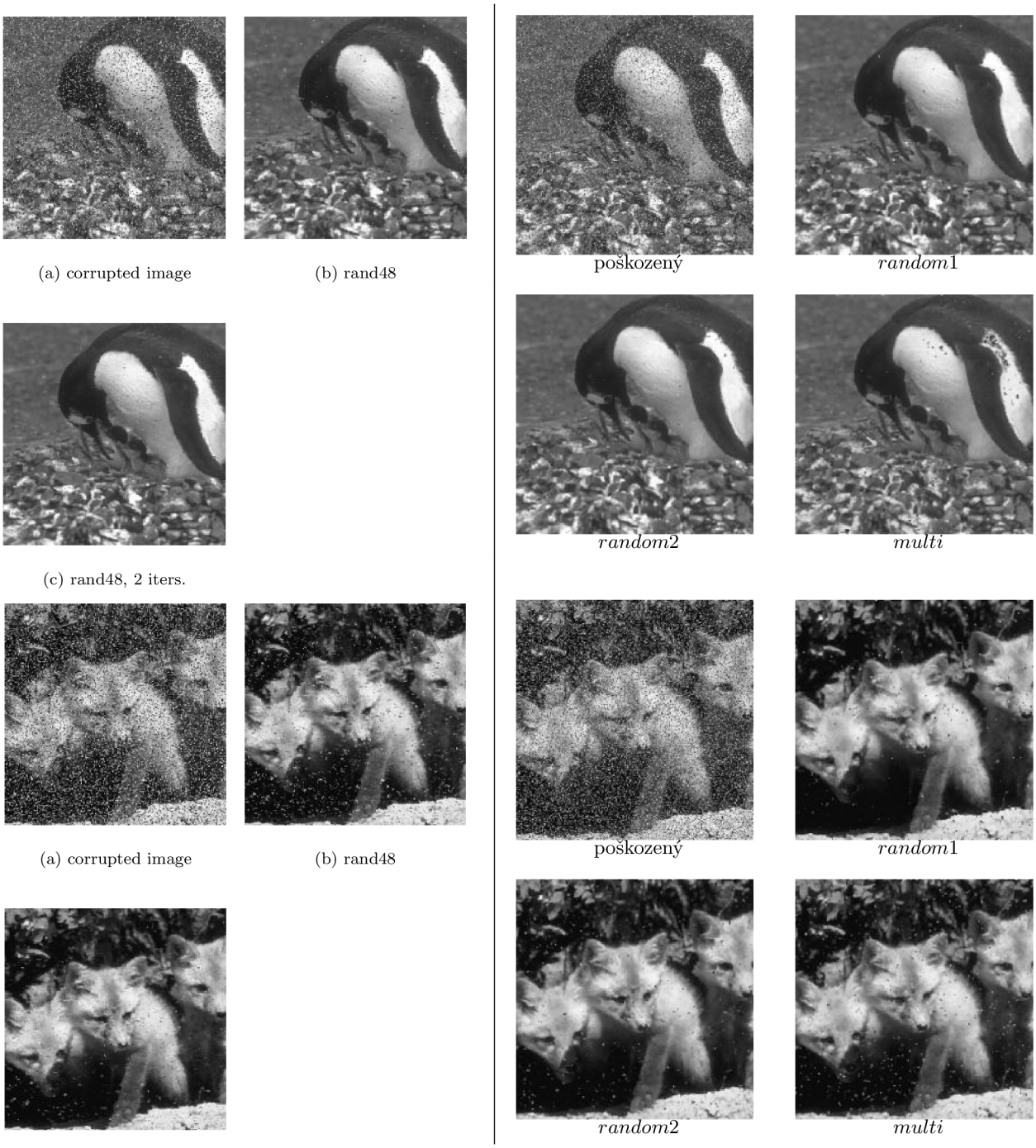


Obrázek 5.31: Souhrnné porovnání kvality podle průměrné hodnoty PSNR z vícekrokové filtrace sady 30 obrazů poškozených náhodným šumem metodou `vasicek2013` (označení *rand*). Převzato z [43]

Mediánové filtry MF 3x3, 5x5 a EMF se u tohoto šumu projevily na rozdíl od šumu *impulse burst* mnohem lépe (viz graf 5.29). Společně s metodou `vasicek2013` dávají zdaleka nejkvalitnější výsledky, nicméně rozdíly v kvalitě filtrace podle hodnot PSNR nejsou tak velké jako u předchozích typů šumů. Navrhovaná metoda se navíc i mezi těmito kvalitními filtry dokázala prosadit, a to včetně filtru *multi*, který už u třetího modelového šumu potvrzuje svou použitelnost s dobrými výsledky. Metody `tourtounis2017` a IDBMF nebyly určeny pro filtraci náhodného šumu, i tak jsou zde zařazeny do porovnání (hlavně pro srovnání víceúčelovosti s filtrem *multi*). Metoda `bidlo2020` byla podobně jako u šumu *impulse burst* přetrénována na náhodném šumu, díky čemuž se dostala velmi blízko k ostatním metodám, co se týče kvality.

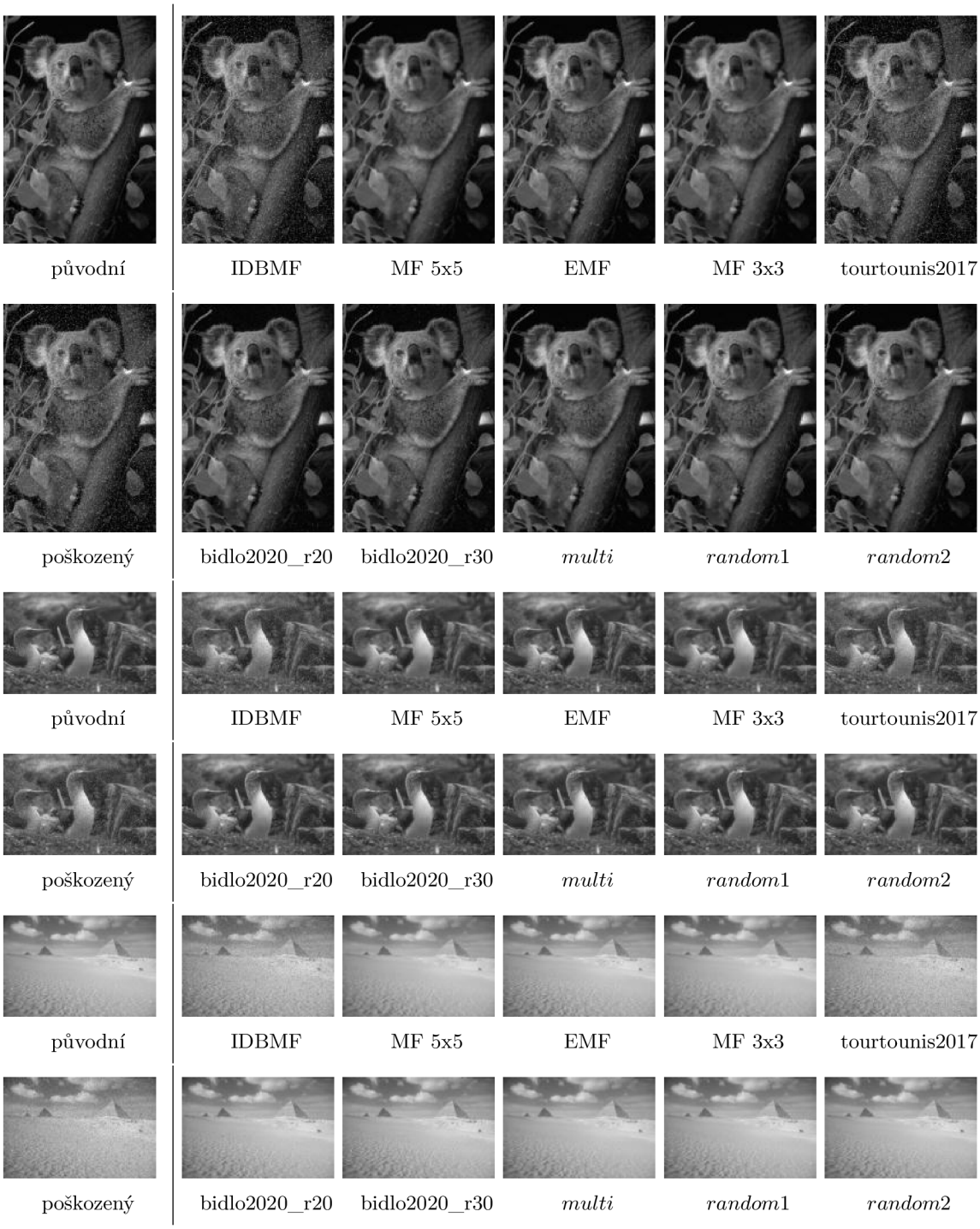
Ukázky a porovnání s metodami jsou opět na různých intenzitách a obrazech, pro metodu `vasicek2013` v obrázcích 5.32, pro ostatní metody v 5.33, 5.34 a 5.35 a ve výřezech 5.36, 5.37 a 5.38. Výsledky filtrace náhodného šumu jsou v některých ohledech podobné jako u šumu sůl a pepř. Mediánové filtry MF 3x3 a 5x5, ač mají vysoké hodnoty PSNR, nezachovávají detaily a hrany a obraz je příliš rozostřený. Zajímavé je u velmi nízkých intenzit zachování detailů přetrénovanou metodou `bidlo2020`, bohužel za cenu ponechání části poškození. Navrhovaná metoda už od nízkých intenzit mírně zahlazuje (pravděpodobný vliv mediánu) a částečně se tak poškození zbavuje. S rostoucí intenzitou šumu se filtrace zhoršuje a u všech metod jsou viditelné zbytky poškození v obraze. Metoda EMF u vyšších intenzit ponechává těchto vizuálně nepříjemných poškození nejméně, ale zase zdůrazněme, že metoda EMF pracuje s větším filtračním oknem (3x3).

Ukázka filtrace na obrazech z metody `vasicek2013`



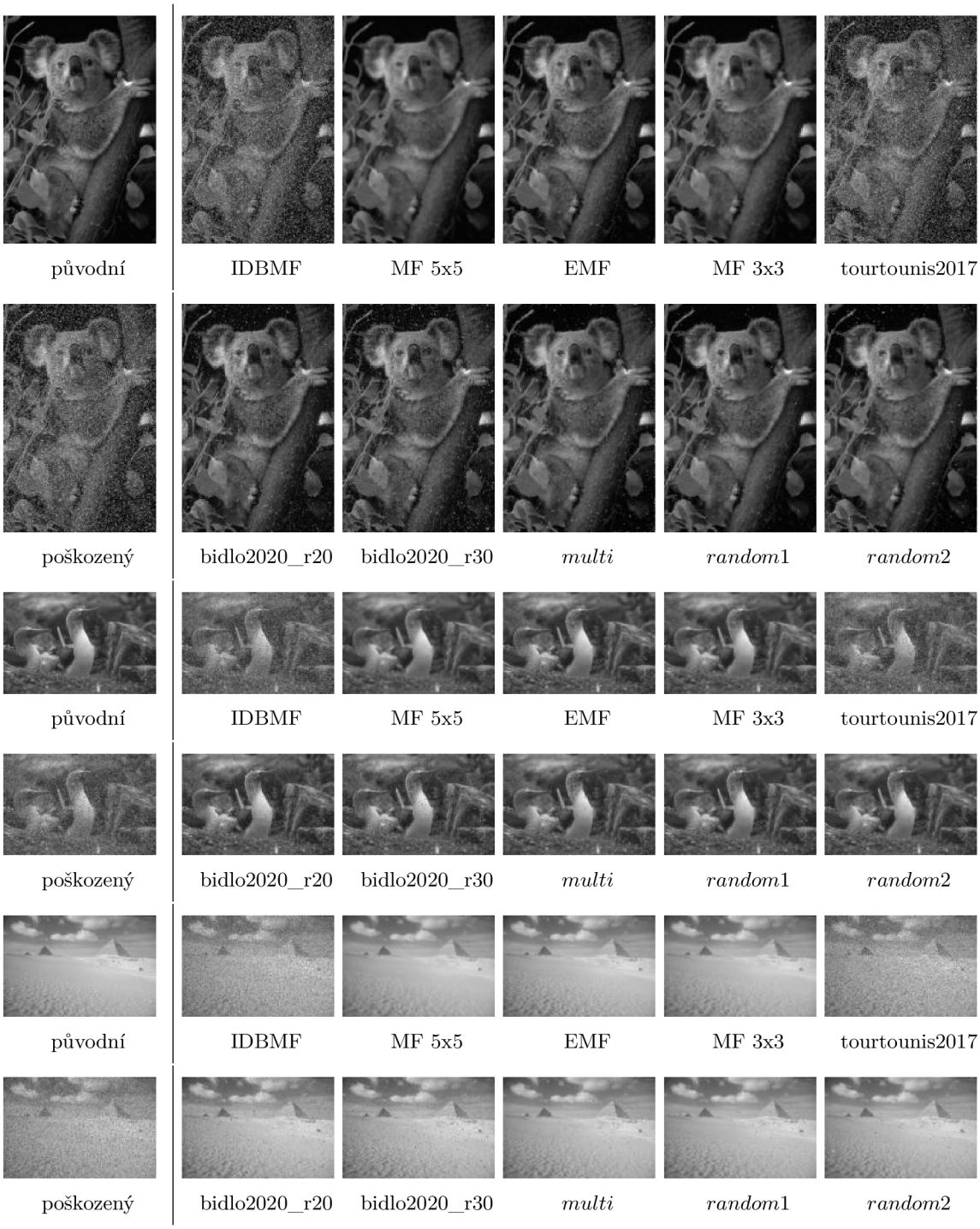
Obrázek 5.32: Porovnání kvality filtrace s metodou `vasicek2013` na obrazech 106202 (tučňák) a 159008 (liščata) poškozenými náhodným šumem s intenzitami 20 % a 35 % respektive. Vlevo (označení vasicek2013, převzato z [43]) je výsledek aplikace filtrů metody `vasicek2013`, vpravo je výsledek aplikace navrhovaných filtrů. Filtry metody `vasicek2013` byly vyhodnoceny po 1 nebo 2 krocích (*2 iters*), navrhované filtry jsou vyhodnoceny po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 10 %



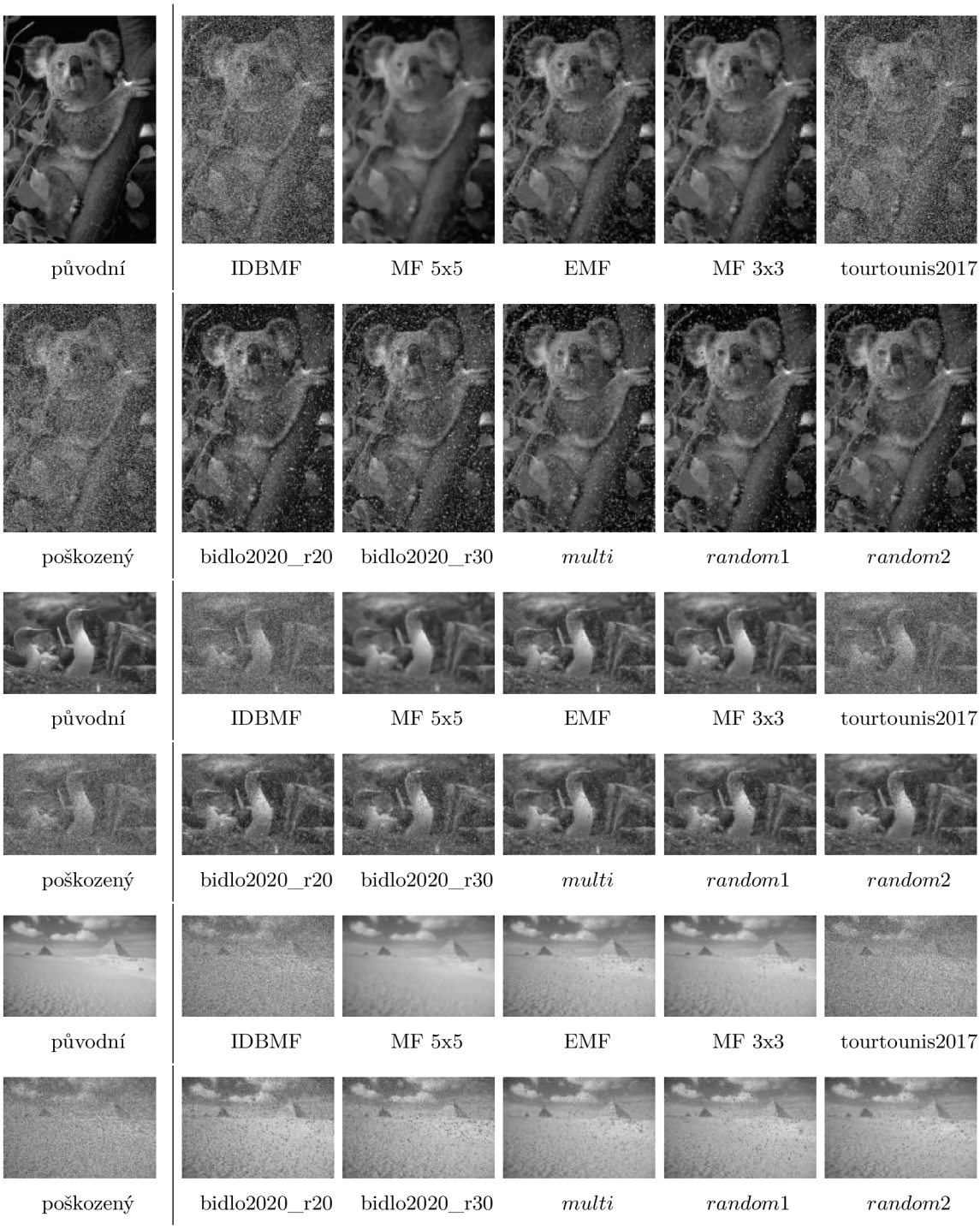
Obrázek 5.33: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené náhodným šumem s intenzitou 10 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR trénované na náhodném šumu, navrženého víceúčelového filtru *multi* a filtry trénované na náhodném šumu *random*1 a *random*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po maximálně 6 krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 2 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 30 %



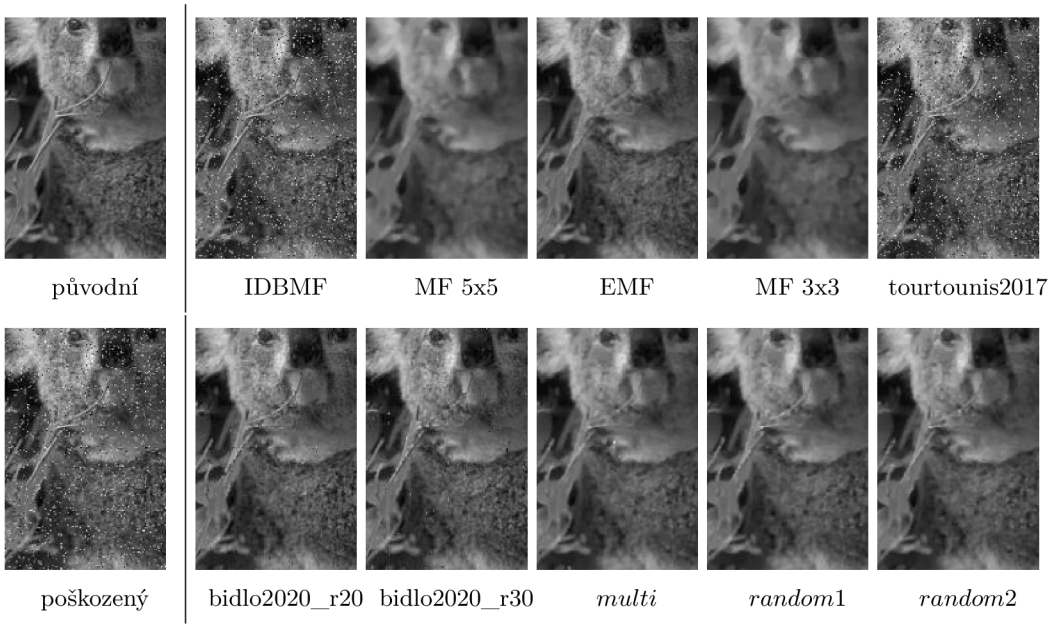
Obrázek 5.34: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené náhodným šumem s intenzitou 30 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR trénované na náhodném šumu, navrženého víceúčelového filtru *multi* a filtry trénované na náhodném šumu *random*1 a *random*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po 6 maximálně krocích aplikace filtru, metoda `tourtounis2017` byla vyhodnocena po 2 krocích a metoda `bidlo2020` a navrhované filtry po 6 krocích.

Ukázka filtrace na obraze poškozeném šumem s intenzitou 50 %

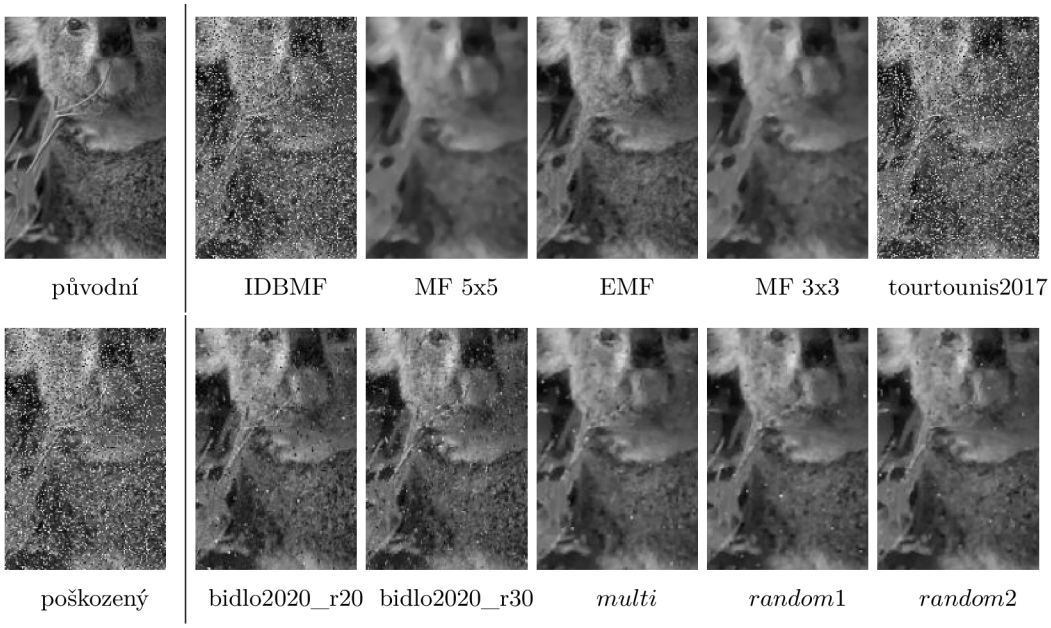


Obrázek 5.35: Obrazy 69015 (koala), 103070 (ptáci) a 260058 (pyramidy) poškozené náhodným šumem s intenzitou 50 % po aplikacích filtrů MF 3x3, MF 5x5, IDBMF, EMF, metody `tourtounis2017`, metody `bidlo2020` s 20 a 30 CMR trénované na náhodném šumu, navrženého víceúčelového filtru *multi* a filtry trénované na náhodném šumu *random*1 a *random*2. Pro konvenční metody byl vybrán nejlepší výsledek dosažený po 6 maximálně krocích aplikace filtru, metoda `tourtounis2017`, metoda `bidlo2020` a navrhované filtry po 6 krocích.

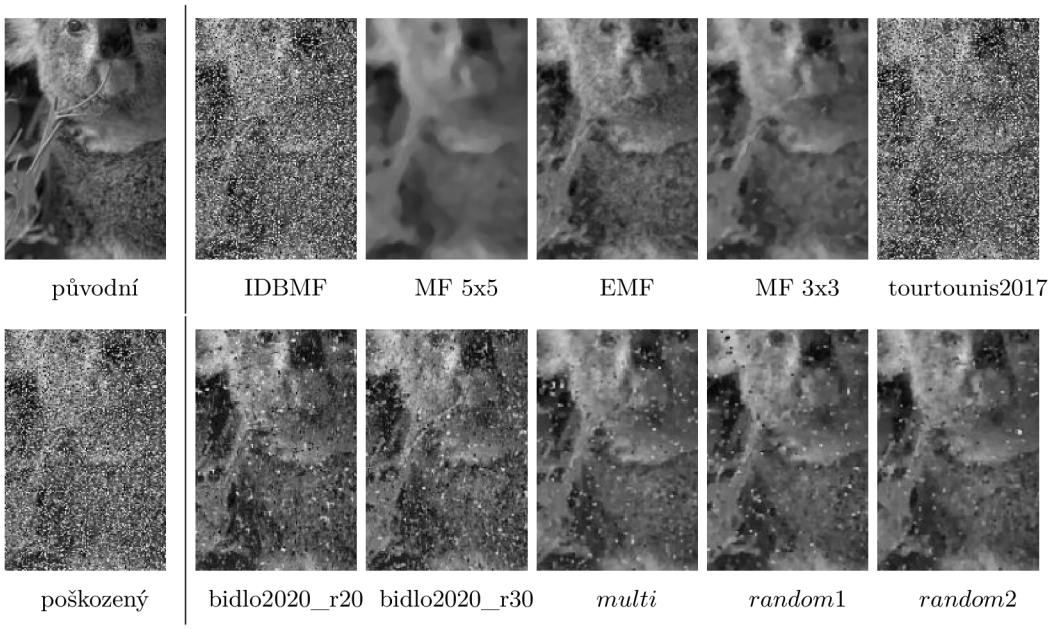
Výřezy z ukázek filtrace



Obrázek 5.36: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.33



Obrázek 5.37: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.34



Obrázek 5.38: Výřez detailu obrazu 69015 (koala) jednotlivých aplikací filtrů z 5.35

Kapitola 6

Závěr

V práci byla prozkoumána a rozšířena metoda z publikace [8] využívající evoluční strategii k získání podmínkových přechodových pravidel, kterými celulární automat filtruje obraz. Podařilo se implementovat původní algoritmus a zreprodukovat jeho experimentální výsledky. Dále byla navržena a implementována modifikace pravé strany podmínkových přechodových pravidel tak, aby došlo ke zkvalitnění filtrace obrazu (viz podkapitola 4.1). Modifikace spočívá v zasazení výběru výpočetní funkce na pravou stranu pravidla, proto bylo provedeno 3234 různých experimentů pro získání vhodné sady výpočetních funkcí. Na základě získaných výsledků byly určeny nejlepší sady velikosti 8 a 4.

V práci byly dále určeny zkoumané modely šumu sůl a pepř, *impulse burst* a náhodný šum, které byly následně použity jak pro trénování, tak pro vyhodnocení a porovnání s jinými existujícími metodami. Experimentovalo se nad standardně zašuměnými obrazy (jeden šum, jedna intenzita šumu) i nad obrazy s různými intenzitami téhož šumu nebo obrazy s různými šumy. Mezi další experimentální parametry patřil počet podmínkových přechodových pravidel a počet kroků celulárního automatu. Bylo provedeno celkem 5460 experimentů s různými kombinacemi těchto parametrů s cílem nalezení nejvhodnějšího nastavení pro získání nejkvalitnějších filtrů. Pro každý modelový šum byly vybrány 2 nejlepší filtry a navíc se podařilo určit jeden víceúčelový filtr, který je schopný kvalitně filtrovat všechny vybrané modelové šumy. Pro získání tohoto filtru se ukázalo jako zásadní trénovat na obraze poškozeném různými typy šumu (sůl a pepř, náhodný šum). Dalším důležitým zjištěním je vlastnost filtrů získaných trénováním na šumu sůl a pepř odstraňovat dobře šum *impulse burst*.

Pro účely zhodnocení výsledků práce byly nastudovány a implementovány metody MF 3x3, MF 5x5, IDBMF, EMF a metoda z publikace [40], viz kapitola 2.4. Další srovnávací metodou, která využívá evoluce k návrhu obrazových filtrů, je metoda z publikace [43]. Navrhovaná metoda a metody určené k porovnání (kromě metody [43]) byly vyhodnoceny na sadě 25 obrazů nad všemi uvažovanými modely šumu. Ze srovnání (včetně praktických ukázek filtrace) vyplývá, že se podařilo výrazně zlepšit původní koncept podmínkových přechodových pravidel a nastavení evoluční strategie. Vybrané nejlepší filtry jsou také schopné kvalitní filtrace v porovnání s ostatními metodami. Víceúčelový filtr navíc na rozdíl od ostatních metod prokazuje schopnost kvalitně filtrovat všechny vybrané modelové šumy. Navrhovaná metoda pracuje pouze s von Neumannovým okolím (5-okolím) a umožňuje rychlou evoluci filtrů (500-1000 generací) pro konkrétní typ šumu. Na běžném počítači je možné získat při vhodném nastavení evoluce kvalitní filtr do pár minut.

Práce byla oceněna odborným panelem a partnerem z průmyslu Cognitechna[1] na studentské konferenci Excel@FIT 2023[2]. Navíc byla i mezi 8 vybranými pracemi, které byly prezentovány během dopoledního programu.

Možnosti dalšího rozšiřování práce jsou zejména v důkladnějším prozkoumání trénování na obraze poškozeném různými šumy (případně i obrazy s reálným poškozením) a zlepšení kvality filtrace náhodného šumu. Dalších zajímavých výsledků by bylo možné dosáhnout změnou sousedství celulárního automatu (Moorovo, 5x5) nebo probádáním jiných výpočetních funkcí.

[1] `https://www.cognitechna.com/en/`

[2] `https://excel.fit.vutbr.cz/vysledky/`

Literatura

[1] Adamatzky, A. a Martínez, G. *Designing Beauty: The Art of Cellular Automata* [https://books.google.cz/books?id=w5iFjgEACAAJ]. Springer International Publishing, 2016. Emergence, Complexity and Computation. ISBN 9783319272696.

[2] Alias, M. S. A., Ibrahim, N. a Zin, Z. M. Salt and pepper noise removal by using improved decision based algorithm. In: *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*. 2017, s. 487–492. DOI: 10.1109/SCORED.2017.8305434.

[3] Bao, Z. a Watanabe, T. Evolutionary design for image filter using GA. In:. únor 2009, s. 1 – 6. DOI: 10.1109/TENCON.2009.5396235.

[4] Berto, F. a Tagliabue, J. Cellular Automata. In: Zalta, E. N., ed. *The Stanford Encyclopedia of Philosophy* [https://plato.stanford.edu/archives/spr2022/entries/cellular-automata/]. Spring 2022. Metaphysics Research Lab, Stanford University, 2022.

[5] Bhanja, S. a Pulecio, J. A review of magnetic cellular automata systems. In:. červen 2011, s. 2373 – 2376. DOI: 10.1109/ISCAS.2011.5938080.

[6] Bidlo, M. On Routine Evolution of Complex Cellular Automata. *IEEE Transactions on Evolutionary Computation*. 2016, sv. 20, č. 5, s. 742–754. DOI: 10.1109/TEVC.2016.2516242.

[7] Bidlo, M. Advances in the Evolution of Complex Cellular Automata. In: Merelo, J. J., Melício, F., Cadenas, J. M., Dourado, A., Madani, K. et al., ed. *Computational Intelligence: International Joint Conference, IJCCI 2016 Porto, Portugal, November 9–11, 2016 Revised Selected Papers* [https://doi.org/10.1007/978-3-319-99283-9_7]. Cham: Springer International Publishing, 2019, s. 123–146. DOI: 10.1007/978-3-319-99283-9_7. ISBN 978-3-319-99283-9.

[8] Bidlo, M. Evolution of Cellular Automata with Conditionally Matching Rules for Image Filtering. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, s. 1–8. DOI: 10.1109/CEC48606.2020.9185767.

[9] Brabazon, A. *Natural computing algorithms*. Berlin: Springer, 2015. Natural computing series. ISBN 978-3-662-43630-1.

[10] Charmouti, B., Junoh, A. K., Azman, W., Wan Muhamad, W. Z. A., Mansor, M. et al. Extended Median Filter For Salt and Pepper Noise In Image. *International Journal of Applied Engineering Research*. Leden 2017, sv. 12, s. 12914–12918.

[11] Codd, E. F. *Cellular Automata.* USA: Academic Press, Inc., 1968. ISBN 0121788504.

[12] Eiben, A. E. a Smith, J. E. *Introduction to Evolutionary Computing.* 2nd. Springer Publishing Company, Incorporated, 2015. ISBN 3662448734.

[13] Fliegel, K. a Svihlik, J. An efficient method of noise suppression in security systems. In:. říjen 2007. DOI: 10.1117/12.733104.

[14] Green, D. Cellular automata models in biology. *Mathematical and Computer Modelling* [https://www.sciencedirect.com/science/article/pii/089571779090010K]. 1990, sv. 13, č. 6, s. 69–74. DOI: https://doi.org/10.1016/0895-7177(90)90010-K. ISSN 0895-7177.

[15] Hansen, N., Arnold, D. a Auger, A. *Evolution Strategies.* Leden 2015.

[16] Ilachinski, A. *Cellular Automata: A Discrete Universe.* World Scientific, 2001.

[17] Kang, B.-H., Lee, D.-H. a Hong, C.-P. Pseudorandom Number Generation Using Cellular Automata. In: Sobh, T., Elleithy, K., Mahmood, A. a Karim, M. A., ed. *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics.* Dordrecht: Springer Netherlands, 2008, s. 401–404. ISBN 978-1-4020-8737-0.

[18] Karasek, J. a Benes, R. Image Filter Design Based on Evolution. In:. Duben 2010, sv. 5.

[19] Kourgli, A. a Oukil, Y. Very High Resolution Satellite Images Filtering. In:. říjen 2013, s. 465–470. DOI: 10.1109/BWCCA.2013.81.

[20] Kumar, N. a Nachamai, M. Noise Removal and Filtering Techniques used in Medical Images. *Oriental journal of computer science and technology.* Březen 2017, sv. 10, s. 103–113. DOI: 10.13005/ojcst/10.01.14.

[21] Kvasnička, V. *Evolučné algoritmy.* 1. vydanie. Bratislava: Slovenská technická univerzita v Bratislave vo Vydavateľstve STU, 2000. Edícia vysokoškolských učebníc. ISBN 80-227-1377-5.

[22] Langton, C. G. Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena* [https://www.sciencedirect.com/science/article/pii/0167278984902562]. 1984, sv. 10, č. 1, s. 135–144. DOI: https://doi.org/10.1016/0167-2789(84)90256-2. ISSN 0167-2789.

[23] Liu, S., Chen, H. a Yang, S. An Effective Filtering Algorithm for Image Salt-Pepper Noises Based on Cellular Automata. In: *2008 Congress on Image and Signal Processing.* 2008, sv. 3, s. 294–297. DOI: 10.1109/CISP.2008.263.

[24] Luke, S. *Essentials of Metaheuristics* [http://cs.gmu.edu/~sean/book/metaheuristics/]. Second. Lulu, 2013.

[25] Menshutina, N. V., Kolnoochenko, A. V. a Lebedev, E. A. Cellular Automata in Chemistry and Chemical Engineering. *Annual Review of Chemical and Biomolecular Engineering* [https://doi.org/10.1146/annurev-chembioeng-093019-075250]. 2020, sv. 11, č. 1, s. 87–108. DOI: 10.1146/annurev-chembioeng-093019-075250. PMID: 32513081.

[26] Miller, J. F. Cartesian Genetic Programming. In: Miller, J. F., ed. *Cartesian Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, s. 17–34. DOI: 10.1007/978-3-642-17310-3_2. ISBN 978-3-642-17310-3. Dostupné z: https://doi.org/10.1007/978-3-642-17310-3_2.

[27] Neumann, J. von. *Theory of Self-Reproducing Automata* [https://www.bibsonomy.org/bibtex/2892b53e0cbc3357711fe3086aeb1194a/idsia]. Champain, IL: University of Illionois Press, 1966.

[28] O'Neill, M. E. *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation* [https://www.cs.hmc.edu/tr/hmc-cs-2014-0905.pdf]. HMC-CS-2014-0905. Claremont, CA: Harvey Mudd College, září 2014.

[29] Pathak, M., Sadawarti, H. a Singh, S. A Technique to Suppress Speckle in Ultrasound Images using Nonlocal Mean and Cellular Automata. *Indian Journal of Science and Technology*. Duben 2016, sv. 9. DOI: 10.17485/ijst/2016/v9i13/80421.

[30] Pavelek, M., Janotková, E. a Štětina, J. *Vizualizační a optické měřicí metody*. FSI VUT. Brno: FSI VUT. Brno, january 2001.

[31] Pourhasanzade, F. a Sabzpoushan, S. H. A cellular automata model of chemotherapy effects on tumour growth: targeting cancer and immune cells. *Mathematical and Computer Modelling of Dynamical Systems* [https://doi.org/10.1080/13873954.2019.1571515]. Taylor & Francis. 2019, sv. 25, č. 1, s. 63–89. DOI: 10.1080/13873954.2019.1571515.

[32] Rothman, D. H. a Zaleski, S. *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics*. Cambridge University Press, 1997. Collection Alea-Saclay: Monographs and Texts in Statistical Physics.

[33] Rozenberg, G., Bck, T. a Kok, J. N. *Handbook of Natural Computing*. 1st. Springer Publishing Company, Incorporated, 2011. ISBN 3540929096.

[34] Sahin, U., Uguz, S. a Sahin, F. Salt and pepper noise filtering with fuzzy-cellular automata. *Computers & Electrical Engineering* [https://www.sciencedirect.com/science/article/pii/S0045790613002929]. 2014, sv. 40, č. 1, s. 59–69. DOI: https://doi.org/10.1016/j.compeleceng.2013.11.010. ISSN 0045-7906. 40th-year commemorative issue.

[35] Sekanina, L., Harding, S. L., Banzhaf, W. a Kowaliw, T. Image Processing and CGP. In: Miller, J. F., ed. *Cartesian Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, s. 181–215. DOI: 10.1007/978-3-642-17310-3_6. ISBN 978-3-642-17310-3. Dostupné z: https://doi.org/10.1007/978-3-642-17310-3_6.

[36] Selvapeter, P. J. a Hordijk, W. Cellular automata for image noise filtering. In: *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. 2009, s. 193–197. DOI: 10.1109/NABIC.2009.5393684.

[37] Shah, A., Bangash, J. I., Khan, A. W., Ahmed, I., Khan, A. et al. Comparative analysis of median filter and its variants for removal of impulse noise from gray scale images. *Journal of King Saud University - Computer and Information Sciences* [https://www.sciencedirect.com/science/article/pii/S131915782030327X]. 2022, sv. 34, č. 3, s. 505–519. DOI: https://doi.org/10.1016/j.jksuci.2020.03.007. ISSN 1319-1578.

[38] Shukla, A. Training cellular automata for image edge detection. *Romanian Journal of Information Science and Technology*. Leden 2016, sv. 19, s. 338–359.

[39] Sikulova, M. a Sekanina, L. Acceleration of Evolutionary Image Filter Design Using Coevolution in Cartesian GP. In: Coello, C. A. C., Cutello, V., Deb, K., Forrest, S., Nicosia, G. et al., ed. *Parallel Problem Solving from Nature - PPSN XII*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 163–172. ISBN 978-3-642-32937-1.

[40] Tourtounis, D., Mitianoudis, N. a Sirakoulis, G. C. Salt-n-pepper noise filtering using cellular automata. *ArXiv preprint arXiv:1708.05019*. 2017.

[41] Trefzer, M. A. a Tyrrell, A. M. *Evolvable Hardware: From Practice to Application*. 1st ed. 2015. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2015. Natural Computing Series. ISBN 9783662446157.

[42] Turing, A. 395Intelligent Machinery (1948). In: *The Essential Turing* [https://doi.org/10.1093/oso/9780198250791.003.0016]. Oxford University Press, Září 2004. DOI: 10.1093/oso/9780198250791.003.0016. ISBN 9780198250791.

[43] Vasicek, Z., Bidlo, M. a Sekanina, L. Evolution of efficient real-time non-linear image filters for FPGAs. *Soft Computing*. Listopad 2013, sv. 17. DOI: 10.1007/s00500-013-1040-8.

[44] Weisstein, E. W. *Totalistic cellular automaton* [https://mathworld.wolfram.com/TotalisticCellularAutomaton.html]. Wolfram Research, Inc., Jan 2003.

[45] Šípek, A. *Genetika populací* [http://www.genetika-biologie.cz/genetika-populaci]. MUDr. Antonín Šípek jr., Aug 2010.

Příloha A

Obsah přiloženého paměťového média

Obsah paměťového média je strukturován takto:

- `experiments_rsm4.zip`: archiv s experimenty s vybranou sadou výpočetních funkcí velikosti 4 ($rsm4$),
- `experiments_rsm8.zip`: archiv s experimenty s vybranou sadou výpočetních funkcí velikosti 8 ($rsm8$),
- `experiments_varianty_rsm4.zip`: archiv s experimenty s nad různými sadami výpočetních funkcí velikosti 4,
- `experiments_varianty_rsm8.zip`: archiv s experimenty s nad různými sadami výpočetních funkcí velikosti 8,
- *`src`*: složka se zdrojovými kódy pro spuštění a ohodnocení experimentů a vytvoření ukázek,
- *`ohodnoceni_experimentu`*: složka s `JSON` soubory s výsledky ohodnocení experimentů,
- *`ohodnoceni_ostatni`*: složka s `JSON` soubory s výsledky ohodnocení metod vybraných do porovnání s navrhovanou metodou,
- *`grafy`*: složka s `PDF` grafy vygenerovanými z výsledků experimentů,
- *`csv_data`*: složka s `CSV` soubory vygenerovanými z výsledků experimentů,
- *`excel_fit`*: složka s materiály pro konferenci Excel@FIT 2023 (abstrakt, A1 plakát, prezentace, video),
- `DIP_text.pdf`: písemná zpráva,
- *`text_src`*: složka se zdrojovým tvarem písemné zprávy.

Příloha B

Manuál

Spuštění a ohodnocování experimentů včetně vytváření ukázek bylo implementováno v jazycích C++[1] a Python[2]. Samotné jádro (CA provádějící filtrace a ES navrhující CMR) je implementováno v C++. Zpracování příkazových argumentů, generování konstant a hlavičkových souborů, kompilace programu s konkrétním nastavením, spuštění evoluce a vyhodnocení filtrace jsou řešeny v jazyce Python. Pro ukládání postupu práce byl využit verzovací nástroj `git`[3]. Pro generování pseudonáhodných čísel byla zvolena knihovna `PCG` [28].

Struktura repozitáře

Repozitář s implementací práce má následující strukturu:

- `runner.py`: skript pro spuštění experimentu,
- `shower.py`: skript pro vygenerování ukázky filtrace,
- `grader.py`: skript pro ohodnocení experimentů,
- `config.py`: konfigurační soubor,
- `constants_generator.py`: pomocný modul pro generování konstant,
- `results_generator.py`: pomocný modul pro shrnutí výsledků experimentu,
- `requirements.txt`: soubor s požadovanými Python knihovnami,
- `README.md`: manuál,
- *`data`*: složka s vygenerovanými nebo referenčními daty,
  - *`data/experiments`*: složka s daty experimentů,
  - *`data/gifs`*: složka s `GIF` ukázkami filtrace,
  - *`data/ref`*: složka s referenčními obrázky,
  - *`data/tmp`*: složka s dočasnými soubory ukázek,
- *`libs`*: složka se zdrojovými soubory knihoven,

[1] `https://cplusplus.com/`
[2] `https://www.python.org/`
[3] `https://git-scm.com/`

- *obj*: složka s objektovými kompilačními soubory,
- *src*: složka se zdrojovými soubory pro CA, CMR a ES,
  - `src/ca.cpp`, `ca.h`: zdrojové soubory implementující funkcionalitu CA,
  - `ea.cpp`, `ea.h`: zdrojové soubory implementující funkcionalitu ES,
  - `output_func.cpp`, `output_func.h`: zdrojové soubory implementující funkcionalitu pravé strany CMR,
  - `constants.cpp`, `constants.h`: zdrojové soubory s konstantními hodnotami,
  - `main.cpp`: hlavní řídicí program,
  - `Makefile`.

Instalace

Požadované nástroje jsou `g++`[4], OpenMP[5], `make`[6] a Python 3.9+. Implementace byla testována na operačním systému Linux (Fedora 37[7], RHEL 8.6[8] a CentOS Linux 7 (Core)[9] na školním serveru *merlin*). Pro spuštění jednotlivých skriptů je také nutné nainstalovat požadované Python knihovny (`matplotlib`, `Pillow`) uvedené v souboru `requirements.txt`.

Příprava virtuálního prostředí a vytvoření složek pro skripty:

```
$ python -m venv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
$ mkdir obj data/experiments data/gifs
```

Spuštění programu

Program umožňuje jak spuštění experimentů, tak jejich vyhodnocení na sadě 25 obrazů univerzity Berkeley. Dále je možné vygenerovat ukázky aplikace filtru na obrazech.

Spuštění experimentu

Spuštěním skriptu `runner.py` se v adresáři *data/experiments* vytvoří složka pro experiment s jeho kódovým označením podle hodnot parametrů. V případě více shodných experimentů je přidáno číslo na konec názvu experimentu. Jsou vygenerovány podmínkové tabulky, poškozené obrazy a další konstantní hodnoty a kód je zkompilován a puštěn paralelně v zadaném počtu běhů. Každý běh ukládá data průběhu evoluce (ve formátu: generace, PSNR nejlepšího dosaženého jedince, chromozom nejlepšího dosaženého jedince) do souboru s názvem `xxx.txt`, kde `xxx` je číselný kód běhu (`000.txt`, `001.txt`, atd.). Po dokončení všech běhů je vygenerován graf průběhu evoluce všech běhů a ukázka filtrace nejlepšího dosaženého filtru z běhů na obraze s nastavením, na kterém byl trénován.

[4] `https://gcc.gnu.org/`
[5] `https://www.openmp.org/`
[6] `https://www.gnu.org/software/make/`
[7] `https://fedoraproject.org/`
[8] `https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux`
[9] `https://www.centos.org/`

Příkaz pro spuštění experimentů:

```
$ python runner.py [--img I] [--noise N] [--damage D] [--steps S] [--rules R]
                   [--generations G] [--runs B] [--rsm M] [--outfuncs O]
```

- `--img I`, trénovací obraz, `I` je název souboru obrazu včetně přípony ve složce *`data/ref`*, např. `koala.jpg`, výchozí hodnota: `lena.png`;
- `--noise N`, trénovací typ šumu, `N` je jedno z vybraných:
  - `salt_pepper` (výchozí hodnota),
  - `random`,
  - `impulse_burst`,
  - `salt_pepper_halved`,
  - `random_halved`,
  - `salt_pepper_random_halved`;
- `--damage D`, trénovací intenzita šumu v procentech, `D` může být:
  - pro šumy `salt_pepper`, `random`, `impulse_burst`: celé číslo z intervalu $\langle 0, 100 \rangle$, výchozí hodnota: 10,
  - pro šumy s koncovkou `_halved` se intenzita skládá z dvou celých čísel z intervalu $\langle 0, 100 \rangle$, např. 1020, 9090, 60 (pro 0 % a 60 %), výchozí hodnota: 10;
- `--steps S`, počet kroků filtrace CA, po kterých se vyhodnocuje funkce fitness, `S` je celé číslo, výchozí hodnota: 3;
- `--rules R`, počet navrhovaných pravidel (CMR) přechodové funkce CA, `R` je celé číslo, výchozí hodnota: 20;
- `--generations G`, počet generací ES, `G` je celé číslo, výchozí hodnota: 2500;
- `--runs B`, počet paralelních běhů ES, `B` je celé číslo, výchozí hodnota: 1;
- `--rsm M`, počet možných výpočetních funkcí na pravé straně pravidla, `M` je buď 4 nebo 8, výchozí hodnota: 8;
- `--outfuncs O`, výběr možných výpočetních funkcí na pravé straně pravidla, `O` je řetězec názvů metod včetně vstupních parametrů oddělených mezerami, výchozí hodnota: `"median(neighs) min(neighs) max(neighs) neighs[0] neighs[1] neighs[3] neighs[4] val"`, další možné funkce jsou:
  - `mean(neighs)`, `half(neighs)`, `major(neighs)`, `minor(neighs)`, `gauss(neighs)`, `bilateral(neighs)`.

Příklady použití:

```
$ python runner.py --img koala.jpg --noise salt_pepper_halved --damage 2060 \
    --steps 2 --rules 15 --generations 500 --runs 5 --rsm 4 --outfuncs \
    "mean(neighs) minor(neighs) neighs[1] major(neighs)"
$ python runner.py --img birds.jpg --noise random --damage 60 --steps 5 \
    --rsm 4 --generations 1000 --rules 50
$ python runner.py --generations 100
```

Vygenerování ukázky

Spuštěním skriptu `shower.py` dojde k aplikaci filtru `RULES` v několika krocích. Po každém kroku je na výstup vypsána hodnota fitness (PSNR), a pokud není zvolena možnost `--all_fitness`, tak je pro každý krok vytvořen obrázek ve složce *`data/tmp`*. Data ve složce *`data/tmp`* jsou při dalším generování ukázek přepsána.

Příkaz pro vygenerování ukázky:

```
$ python shower.py [--best] [--gif] [--skip_compile] [--all_fitness]
                   [--img I] [--noise N] [--damage D] [--steps S]
                   [--rsm M] [--outfuncs O] RULES
```

- `--best`, možnost pro vytvoření ukázky aplikace jednoho z nejlepších filtrů, výběr filtrů se zadává jeho názvem do parametru `RULES`, možnosti: "salt1", "salt2", "impulse1", "impulse2", "random1", "random2", "multi";
- `--gif`, možnost pro vygenerování `GIF` ukázky z jednotlivých kroků filtrace CA, výsledek je uložen ve složce *`data/gif`*;
- `--skip_compile`, možnost pro vynechání kompilace, v tomto případě není využit žádný z nastavitelných parametrů kromě `RULES`;
- `--all_fitness`, možnost pro vygenerování hodnot fitness pro intenzity 10-90 % bez vygenerování obrazových ukázek, parametr `--damage` není v tomto případě zpracovaný a je doporučeno použít pouze typ poškozením celého obrazu (`salt_pepper`, `random` nebo `impulse_burst`);
- `--corr_img`, možnost pro aplikaci filtru na již poškozený obraz, volitelné parametry `img`, `noise` a `damage` v tomto případě nejsou zpracovány a výstupní hodnoty fitness (PSNR) nejsou relevantní (není k dispozici původní nepoškozený obraz);
- `--img I`, `I` je název souboru obrazu včetně přípony ve složce *`data/ref`*, např. `koala.jpg`, výchozí hodnota: `lena.png`;
- `--noise N`, typ šumu, `N` je jedno z vybraných:
  - `salt_pepper` (výchozí hodnota),
  - `random`,
  - `impulse_burst`,
  - `salt_pepper_halved`,
  - `random_halved`,
  - `salt_pepper_random_halved`;
- `--damage D`, intenzita šumu v procentech, `D` může být:
  - pro šumy `salt_pepper`, `random`, `impulse_burst`: celé číslo z intervalu $\langle 0, 100 \rangle$, výchozí hodnota: 10,
  - pro šumy s koncovkou `_halved` se intenzita skládá z dvou celých čísel z intervalu $\langle 0, 100 \rangle$, např. 1020, 9090, 60 (pro 0 % a 60 %), výchozí hodnota: 10;

- `--steps S`, počet kroků filtrace CA, po kterých se vyhodnocuje funkce fitness, `S` je celé číslo, výchozí hodnota: 7;
- `--rsm M`, počet možných výpočetních funkcí na pravé straně pravidla, `M` je buď 4 nebo 8, výchozí hodnota: 8;
- `--outfuncs O`, výběr možných výpočetních funkcí na pravé straně pravidla, `O` je řetězec názvů metod včetně vstupních parametrů oddělených mezerami, výchozí hodnota: `"median(neighs) min(neighs) max(neighs) neighs[0] neighs[1] neighs[3] neighs[4] val"`, další možné funkce jsou:
  - `mean(neighs)`, `half(neighs)`, `major(neighs)`, `minor(neighs)`, `gauss(neighs)`, `bilateral(neighs)`;
- `RULES`, **povinný parametr**, filtr v celočíselné reprezentaci CMR (chromozom jedince ES) v uvozovkách (posloupnost musí mít délku dělitelnou 6), např. "10 59 150 647 999 145 2 0 ... 569 87";

Příklady použití:

```
$ python shower.py --img koala.jpg --noise salt_pepper_halved --damage 2060 \
    --steps 10 --gif --best "salt1"
$ python shower.py --rsm 4 --steps 5 "713 890 670 774 893 120 822 748 420 \
787 489 162 836 300 182 765 312 42 824 703 928 112 669 237 262 723 89 252 \
11 987 479 765 982 722 546 539 685 300 822 324 508 128 287 519 858 713 631 \
797 641 608 788 35 854 181 809 834 551 335 158 872"
$ python shower.py --all_fitness --steps 7 --noise impulse_burst "901 895 \
88 750 889 250 272 587 1022 739 939 1008 213 112 161 848 683 1292 785 137 \
5 649 698 938 763 297 1021 384 1010 1016 766 739 1016 745 417 48 1002 350 \
1016 779 1015 400 768 578 165 71 971 1750 105 531 220 342 916 1308 353 430 \
1016 465 441 1753 47 879 150 861 213 1514 375 725 82 350 346 63 788 290 \
166 278 757 851 665 684 177 718 438 1638 758 737 1007 578 698 831"
```

Ohodnocení experimentů

Experimenty jsou paralelně[10] ohodnoceny na intenzitách 10-90 % na daném typu šumu na sadě 25 obrazů univerzity Berkeley (ve složce *`data/ref`*). Ohodnocení je provedeno pro každý běh experimentu a výsledky jsou pro experiment zprůměrovány podle počtu běhů a vyhodnocených obrazů. Pro každou intenzitu jsou dále nalezeny nejkvalitnější filtry v daném experimentu. Data jsou uložená do souboru formátu `JSON`.

Příkaz pro ohodnocení experimentů:

```
$ python grader.py
Please write regex to filter folders [.*]: ...
Please write noise type to be graded on [salt_pepper]: ...
```

Před spuštěním vyhodnocení se skript dotáže na regulární výraz, podle kterého budou vybrány experimenty k ohodnocení, a typ šumu, na kterém má ohodnocovat. Zadaný regulární výraz musí být ve tvaru, který přijímá Python knihovna `re`. Typ šumu může být

[10] Počet paralelních procesů je určen konstatou `ASYNC_PROCS` přímo ve zdrojovém souboru `grader.py` a měla by být upravena podle možností zařízení, na kterém je program spuštěn.

`salt_pepper`, `impulse_burst` nebo `random`. Výrazy v závorkách u dotazování jsou výchozí hodnoty, které se použijí v případě, že nebude zadán žádný vstup. Ohodnocení pracuje pouze s vybranými sadami $rsm4$ a $rsm8$.

Struktura výsledného souboru `JSON` s ohodnocením experimentů je pak následující:

```
{
    "nazev_experimentu": {
    "best_run_rule": [
        [
            nejlepsi_PSNR_na_10%_sumu,
            nazev_souboru_behu,
            CMR_nejlepsiho_filtru_na_10%_sumu
        ],
        ...,
        [
            nejlepsi_PSNR_na_90%_sumu,
            nazev_souboru_behu,
            CMR_nejlepsiho_filtru_na_90%_sumu
        ]
    ],
    "run_data": [
        [
            prumerne_PSNR_na_10%_sumu_krok_1, prumerne_PSNR_na_10%_sumu_krok_2,
            prumerne_PSNR_na_10%_sumu_krok_3, prumerne_PSNR_na_10%_sumu_krok_4,
            prumerne_PSNR_na_10%_sumu_krok_5, prumerne_PSNR_na_10%_sumu_krok_6
        ],
        ...,
        [
            prumerne_PSNR_na_90%_sumu_krok_1, prumerne_PSNR_na_90%_sumu_krok_2,
            prumerne_PSNR_na_90%_sumu_krok_3, prumerne_PSNR_na_90%_sumu_krok_4,
            prumerne_PSNR_na_90%_sumu_krok_5, prumerne_PSNR_na_90%_sumu_krok_6
        ]
    ],
    "run_count": pocet_behu
    },
    ...
}
```