

tel.: **487 833 123**

fax: **487 833 101**

email: **sps@sps-cl.cz**

web: **www.sps-cl.cz**

## MATURITNÍ PRÁCE

### REKONSTRUKCE OBRAZU UMĚLOU INTELIGENCÍ

Studijní obor: **18-20-M/01 INFORMAČNÍ TECHNOLOGIE**

Autor:

**Jaroslav Holaj**

Podpis:

Vedoucí práce:

**Ing. Michal Michna**

Třída: **4.D**

Školní rok: **2021/2022**





### ZÁVAZNÁ PŘIHLÁŠKA K ŘEŠENÍ MATURITNÍ PRÁCE

Příjmení a jméno žáka: Holaj Jaroslav  
2021/2022

Třída: 4.D

Školní rok:

Téma: Rekonstrukce obrazu umelou inteligencí

Způsob zpracování a pokyny k obsahu a rozsahu maturitní práce:

Vedoucí práce (VP): Michna Michal, Ing.

Podpis VP:

Licenční ujednání:

1. Ve smyslu § 60 autorského zákona č. 121/2000 Sb. poskytuji Střední průmyslové škole, Česká Lípa, Havlíčkova 426, příspěvková organizace výhradní a neomezená práva (§46 a §47) k využití mé maturitní práce.
2. Bez svolení školy se zdržím jakéhokoliv komerčního využití mé práce.
3. V případě komerčního využití práce školou obdrží žák – autor práce odměnu ve výši jedné třetiny dosaženého zisku.
4. Pro výukové účely a prezentaci školy se vzdávám nároku na odměnu za užití díla.

v České Lípě dne: 29. 11. 2021

Podpis žáka:

Termín odevzdání:

Kritéria hodnocení: 1. za vypracování od vedoucího práce

2. za vypracování od oponenta práce

3. obhajoba práce bude hodnocena komisí.

Výsledné hodnocení bude rozhodnutím komise s přihlédnutím k hodnocení bodů 1. až 3.



**Požadavky:** Žák odevzdá práci včetně příloh elektronicky v pdf souboru dle vzoru vedoucímu práce.

**Vyjádření třídního učitele:**

Doporučuji - **Nedoporučují**<sup>1)</sup> konat MP.

**Vyjádření ředitele školy:**

Ředitel školy stanovil délku obhajoby maturitní práce na 20 minut.

Povoluji - **Nepovoluji**<sup>1)</sup> konat MP

1) nehodící se škrtněte

Příloha:

1. Osnova práce



## Osnova Maturitní práce

Příjmení a jméno žáka: Jaroslav Holaj

Třída: 4.D

Téma: Rekonstrukce obrazu umělou inteligencí

Vedoucí práce: Michna Michal, Ing.

Cílem práce je prezentovat současné možnosti rekonstrukce obrazu umělou inteligencí. Vyzkoušet různé nástroje, porovnat výsledky a vytvořit z praktické části alespoň 5 minutové video.

Osnova:

1. Úvod
2. Teoretická část
  - a. Různé metody (5x) použitelné pro rekonstrukci digitálního obrazu
  - b. Aktuální možnosti rekonstrukce pomocí umělé inteligence (UI)
  - c. Metriky používané pro měření kvality výsledné rekonstrukce
3. Praktická část
  - a. Vyzkoušet 5 různých nástrojů pro rekonstrukci pomocí UI, porovnat
  - b. Vytvořit program pro zjištění kvality rekonstruovaných obrazů
  - c. Implementovat vybraný algoritmus, případně navrhnut zlepšení
4. Závěr

*V odevzdání práci musí být popsána prezentovaná praktická část práce.*

## **Licenční ujednání**

Ve smyslu zákona č. 121/2000 Sb., O právu autorském, o právech souvisejících s právem autorským, ve znění pozdějších předpisů (dále jen autorský zákon) jsou práva k maturitním nebo ročníkovým pracím následující:

**Zadavatel** má výhradní práva k využití práce, a to včetně komerčních účelů.

**Autor** práce bez svolení zadavatele nesmí využít práci ke komerčním účelům.

**Škola** má právo využít práci k nekomerčním a výukovým účelům i bez svolení zadavatele a autora práce.

## **Prohlášení**

Prohlašuji, že jsem svou ročníkovou práci vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze práce jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu s autorským zákonem.

V České Lípě dne datum ..... .

Jméno a příjmení autora

## **Poděkování**

Rád bych poděkoval panu Ing. Michalovi Michnovi za odbornou konzultaci během práce, za rady a za jeho ochotu s probíráním a vedením mé ročníkové práce.

## **Anotace**

Ve své práci jsem se zabýval oblastí restaurování obrazu pomocí filtrů a neuronových sítí. Cílem mé práce bylo vytvořit balíček měřících metod a následně implementovat a případně pokusit se vylepšit mnou vybraný algoritmus pro obnovu obrazu a odstranění degradujících elementů. Nadále mým úkolem bylo vysvětlení funkcionality jednotlivých způsobů pro rekonstrukci obrazu. Pro splnění cíle jsem využil svých znalostí programování a instalování balíčků v jazyce Python v prostředí WSL.

## **Klíčová slova**

Šum;Rozmazání;Inpainting;Metriky

## **Annotation**

In my work I have been working in the field of image restoration using filters and neural networks. The aim of my work was to create a package of measurement methods and then implement and possibly try to improve my chosen algorithm for image restoration and removal of degrading elements. Furthermore, my task was to explain the functionality of each method for image reconstruction. To accomplish this goal, I used my knowledge of programming and installation of Python packages in the WSL environment.

## **Keywords**

Noise;Blur;Inpainting;Metrics

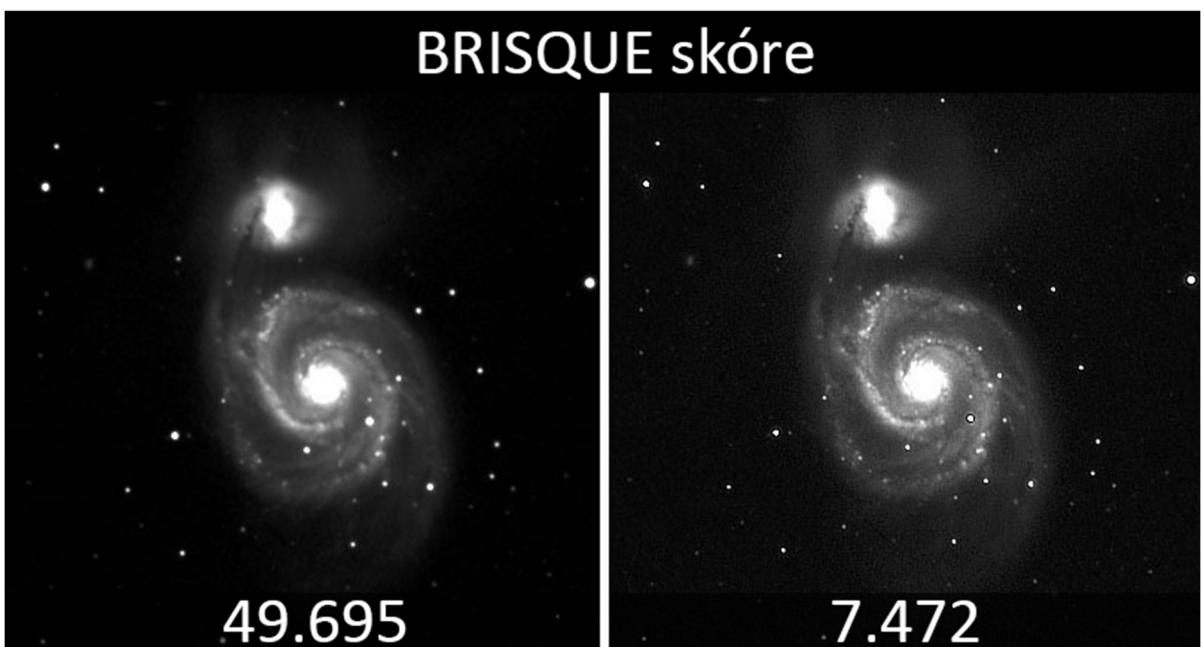
# Obsah

1	Úvod.....	10
2	Teoretická část .....	11
2.1	Metody použitelné pro rekonstrukci digitálního obrazu .....	11
2.1.1	Image Deblurring .....	11
2.1.2	Filtrování šumu.....	14
2.1.3	Obarvování obrazu.....	17
2.1.4	Image Inpainting .....	18
2.1.5	Zvyšování rozlišení.....	19
2.2	Metriky používané pro měření kvality výsledné rekonstrukce .....	20
2.2.1	PSNR .....	20
2.2.2	SSIM .....	21
2.2.3	Butteraugli .....	24
2.2.4	BRISQUE .....	25
3	Praktická část .....	27
3.1	Různé nástroje pro rekonstrukci pomocí UI .....	27
3.1.1	MPRNet .....	27
3.1.2	LAMA.....	28
3.1.3	SwinIR .....	30
3.1.4	BOPBTL .....	31
3.1.5	MIRNet-TFJS .....	32
3.2	Program pro zjištění kvality rekonstruovaných obrazů .....	33
3.2.1	Důležité faktory hodnocení.....	33
3.2.2	Kód.....	34
3.3	Implementace algoritmu / jeho vylepšení .....	37
3.3.1	Instalace .....	37
3.3.2	Spuštění programu a výsledky .....	40

3.3.3	Vylepšení .....	41
4	Závěr .....	43
	Použitá literatura .....	44
5	Seznam obrázků a tabulek .....	48
6	Přílohy .....	50

# 1 ÚVOD

Oblast rekonstrukce obrazu se začala převážně rozvíjet v období kosmických programů Spojených států a bývalého Sovětského svazu v 50. a počátku 60. let. Díky výzkumu v tomto odvětví jsme získali čistší fotografie Země a sluneční soustavy. Nicméně snímky získané z těchto závodů byly mnohokrát degradovány vlivem prostředí či vibrací uvnitř stroje. Degradace tudíž byla velkým problémem vezmeme-li v úvahu, kolik pořízení těchto fotek peněžně stálo.



Obrázek 1: Ukázka zhoršující se kvality se zvyšujícím se BRISQUE skóre [1]

Na těchto fotografiích například vidíme zaznamenanou galaxii, která byla později zpracována pomocí dekonvolučního filtru. Je jasné, že rekonstrukce obrazu je hojně využívána v astronomii, ale co jiné oblasti? Zbavení šumu a různých vad obrazu je velkou součástí například v medicíně (odstranění vad, které se můžou stát během procesu akvizace) či v kriminalistice (filtrování kamerových záznamů).

## 2 TEORETICKÁ ČÁST

### 2.1 Metody použitelné pro rekonstrukci digitálního obrazu

Bohužel neexistuje jediná, univerzální metoda pro opravy vad obrazu. Každá metoda je vhodná na jiné vady a na jistou míru poškození. Toto odvětví se neustále vyvíjí a přináší nové poznatky a způsoby, proto si zkusíme některé způsoby představit.

#### 2.1.1 Image Deblurring

Jedná se o proces odstranění rozostření obrazu neboli obnovení základního ostrého obrazu z rozmazaného vstupu. Přítomnost rozostření v obrazech má řadu důvodů. Nejviditelnější a nejpozoruhodnější je fotoaparát, který je rozostřený nebo rozmazaný v důsledku pohybu – at' už fotoaparátu nebo fotografovaného objektu.

Z uměleckého hlediska je rozmazání obrazu někdy žádané, ale v nejběžnějších situacích rozmazaný efekt spíše kazí cenné obrazové informace. Ku příkladu rychle jedoucí auto zachycené sledovacím systémem, třes ruky při zmáčknutí spouště či zaostření čočky pouze na jednu oblast. Jedná se tedy o nejčastější vadu fotografií. Jak se ovšem této vady zbavit? V odvětví Image Deblurringu existuje mnoho metod, jak se tohoto kazu zbavit. Každá metoda je něčím unikátní, jak už svým výkonem, tak svými neduhy. [15]



Obrázek 2: Příklad deblurringu SPZ [2]

### 2.1.1.1 Zostření filtry

Inverzní filtrování je technika pro dekonvoluci. To znamená, že když je obraz rozmazán známým „lowpass“ filtrem, je možné obnovit obraz právě filtrem. Tato metoda je však velmi citlivá na šum. Nicméně přístup „snížení“ jednoho degradujícího elementu nám umožňuje kombinování metod. **Wienerův filtr** je nejvíce optimální kompromis mezi inverzním filtrováním a vyhlazováním šumu. [15] Filtr minimalizuje střední kvadratickou chybu a získává z něho lineární odhad původního obrazu. Má za cíl spočítat statistický odhad neznámého signálu pomocí souvisejícího signálu jako vstupu a filtrováním tohoto signálu vytvořit právě odhad jako výstup. Známý signál může například být délka bluru v pixelech a úhel bluru theta ve směru hodinových ručiček. [16] Pro co nejlepší výsledek je tedy nutné znát hodnoty bluru.



Obrázek 3: Blur obrázku se vstupními hodnotami: délka 45px; theta 15 [3]

Dalším problémem v oblasti deblurringu je prstencový efekt, jinak zvaný, „ringing effect“. Jedná se o rušivé signály v blízkosti ostrých hran. Efekt je způsoben zkreslením nebo ztrátou vysokofrekvenčních informací v obraze. [19]



Obrázek 4: Příklad zesíleného ringing effectu [3]

Tomuto efektu lze předejít správným nastavením filtru. Je především způsoben ztrátou vysokofrekvenčních informací v obrazu. Právě proto se vyskytuje převážně u hran. Nejčastěji se ale také objevuje u různých kompresí.

I přes tyto nedostatky je metoda efektivní a hojně se používá ve vědních oborech, jako například v medicíně, pro zobrazení snímků magnetické rezonance. Je vhodná i pro zobrazení focených digitálních snímků, ale bohužel je metoda těžce zautomatizovatelná právě kvůli bruteforcu parametrů. [19]



Obrázek 5: Příklad zobrazení Wiener filtrem [3]

### 2.1.1.2 Zobrazení umělou inteligencí

Další nově zpopularizovanou metodou je právě zobrazení pomocí neuronové sítě. Focení pomocí mobilních telefonů se za poslední léta rapidně zlepšilo, jedná se o jednu z hlavních funkcí. Malá konstrukce telefonů znemožňuje do mobilních telefonů instalaci silných objektivů. To má vliv na kvalitu zachyceného světla. Bohužel, s menším množstvím světla mají telefony, i oproti jedno-čočkovým fotoaparátům horší výkon, pokud jde o test kvality obrazu. K jeho vylepšení se tedy používá zpracování obrazu. Tento přístup se v posledních letech stal dosti populární, jelikož výsledky jsou více než slibné. Výkonnější počítače umožňují výzkumníkům spouštět o mnoho větší modely v kratším čase a s rostoucím počtem mobilních zařízení, fotoaparátů a dalších médií je snazší najít větší soubory dat. Dokáží se naučit a modelovat nelineární a složitá spojení, které by mnoho filtrů praktikovalo jen s obtíží a nutným zásahem uživatele. [17]

Cílem těchto algoritmů je prediktivní učení pod dohledem (s učitelem) a naučit se, jak mapovat neznámý vstup  $x$  na výstup  $y$ . Trénovací data  $x$  může být například matice barevných hodnot pixelů v obrázku. Těmto datům obecněji říkáme rysy.

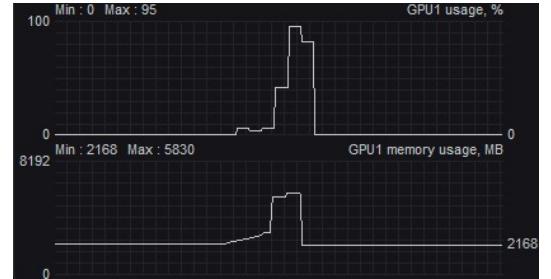
Podobně samotným výstupem může být také vícerozměrné pole (u zobrazení stejně-rozměrné pole obrazu, jako vstup  $x$ ).

Tento způsob má jasné výhody v procesu automatizace, jelikož parametry rozmazání nemusíme jakkoliv upravovat a program se o vše postará sám.



Obrázek 6: SSIM měření kvality obrazu [4]

Největší nevýhodou tohoto přístupu je náročnost na hardware. Digitální obrazy mají mnoho dalších parametrů kromě samotné velikosti, které je nutno uložit, když je aplikován bitová hloubka, do paměti. Většina pracovních stanic nemá k dispozici dostatečnou velikost VRAM schopné uložit a zpracovat tak velkou matici dat, kterou potřebujeme.



Obrázek 7: Využití GPU a VRAM

### 2.1.2 Filtrování šumu

Obecně řečeno, šum je statistická odchylka měření vytvořená náhodným procesem. Ve zpracování obrazu se šum vyskytuje jako náhodný artefakt zrnité struktury pokrývající obraz. Šum v obrazu je vedlejším efektem nepravidelného kolísání signálu, úkolem tedy je vytvořit co nejsilnější signál s minimálním množstvím šumu vedle něj. To se ovšem ukazuje jako problém, zejména v situaci slabého osvětlení, kdy je signál již tak nízký. Jak již bylo řečeno, šum je náhodná složka a z toho lze vyvodit, že lze obraz pro odstranění šumu průměrovat. Tím se náhodné složky zbavíme. Metoda je to ovšem nevhodná, kvůli rozmazanosti obrazu, za to je ale nejjednodušší. Častěji se tedy využívá metoda filtru nebo konvolučních sítí. [19]

Mezi nejzákladnější typy šumu řadíme:

Tabulka 1: Typy šumu

Gaussovský	Hustota pravděpodobnosti šumu má Gaussovo rozdělení.
Sůl a pepř	Impulsní šum, kde náhodné veličiny nabývají tří hodnot. (Bílá, černá, nemění se.). Pravděpodobnost každé složky poté ovlivňuje intenzitu šumu.

Každý z těchto typů se vyskytuje na jiných typech obrazu a je nutné k nim přistupovat jinak.

### 2.1.2.1 Filtrování šumu filtry

Důležitou vlastností dobrého modelu je, že by měl zcela odstranit šum a zachovat zároveň hrany v obrazu. Tradičně existují dva typy modelů, lineární a nelineární model. Hlavní výhodou lineárního modelu je rychlosť. Na úkor toho ale nejsou schopny zachovat hrany šumu efektivním způsobem. V několika případech je nezbytné analyzovat typ šumu, kde struktura a vlastnosti daného signálu mohou dosti napomoci pro vhodný výběr filtrační metody. [19]

K filtrování **Gaussovského šumu** využíváme **Gaussovské rozostření**. Po aplikaci filtru je viditelné zmenšení šumu, za to ale značně degradované detaily obrazu.



Obrázek 8: SSIM hodnoty Gaussovské filtrace [5]

Naopak, jelikož se jedná o naprostě jiný typ šumu, se pro **Sůl a pepř šum** hodí filtr kompletně jiný. V minulém příkladu jsme využívali Gaussovské křivky, zde využijeme **Mediánový filtr**.



Obrázek 9: SSIM hodnoty Mediánového filtrování na Sůl a pepř šumu [5]

Hlavní myšlenkou je procházet signál položku po položce a každou položku poté nahradit mediánem sousedních položek. Tvar výběrového okna bývá nejčastěji čtverec, ovšem můžeme použít jakýkoliv tvar.

### 2.1.2.2 Filtrování šumu umělou inteligencí

Přestože se neuronové sítě rychle rozvíjí, nemusí být nutně efektivním způsobem řešení tohoto problému. Hlavním důvodem je to, že v reálných procesech odstranění šumu chybí dvojice obrazů pro trénink. Všechny stávající metody denoisingu jsou trénovány pomocí simulovaných zašuměných dat vytvořených přidáním pravidelného Gaussova šumu k čistým obrazům. I přesto tvoří významný pokrok v oblasti filtrování šumu.

Cílem metody je „naslepo“ odstranit šum a přiblížit se co nejvíce skutečnosti. Naslepo znamená, že jakákoli architektura umělé inteligence, kterou vytvoříme, by se měla naučit rozložení šumu v obrazech a odstranit jej. Jako vždy, výsledky závisí na typu dat, které modelu poskytneme.

Jak již víme, obraz má 3 kanály (RGB) pro každý pixel. Každá barva je poté reprezentována, například, 8bit číslem v rozmezí 0-255. Dozvěděli jsme se, že šum je náhodné kolísání pixelů. Jinými slovy, určité pixely v RGB kanálech jsou poškozené. Pro jejich obnovení musím hodnoty pixelů opravit. Tím docílíme tak, že algoritmus bude „předpovídat“ skutečnou hodnotu poškozeného pixelu v rozmezí 0-255. [20]



Obrázek 10: SSIM odstranění šumu pomocí MPRNet [5]

### 2.1.3 Obarvování obrazu

Automatizovaná kolorizace snímků je předmětem mnoha výzkumů zabývajících se počítačovým viděním. Má široké využití z hlediska estetiky a restaurování snímků. Jedná se o sofistikovaný úkol, který vyžaduje předchozí znalosti o obrazu a ruční úpravy, aby bylo dosaženo výsledku bez artefaktů. Navíc, objekty mohou být v různých barvách, tudíž neexistuje jedinečné řešení problému. Existují dva hlavní způsoby, jak k problému přistoupit. První vyžaduje, aby uživatel přiřadil barvy k některým oblastem a rozšíří informaci na celý obraz. Druhý způsob se algoritmus snaží naučit barvu každého pixelu z barevného obrazu s podobným obsahem a aplikuje jej. Nejvíce se používá metoda druhá s kombinací konvoluční neuronové sítě (CNN). Ta se skládá z několika malých výpočetních jednotek, které zpracovávají pouze části vstupního obrazu způsobem dopředné mechaniky. Což znamená, že spojení mezi uzly nevytváří cyklus. Každá vrstva je výsledkem použití různých obrazových filtrů, z nichž každý extrahuje určitou vlastnost vstupního obrazu, na předchozí vrstvu. [21]

Každá vrstva tak může obsahovat užitečné informace o vstupním obrazu na různých úrovních abstrakce a s rozvojem grafických procesorů lze dosáhnout pozoruhodných výsledků.



Obrázek 11: Obarvení obrazu pomocí DeepAI [6]

#### 2.1.4 Image Inpainting

Proces vyplňování obrazu má kořeny už v klasických uměleckých dílech, kdy určitá část obrazu byla nenávratně poškozena. Umělci poté „domalovávali“ chybějící kus dle své představy. V současné době se používá mnoho programů, které dokáží rekonstruovat chybějící nebo poškozená místa digitálních fotografií i videí. Přístupy související s inpaintingem obrazů lze rozdělit do dvou kategorií. Metody založené na „**políčkování**“ a na **difúzi**. [22] Kde metoda políčkování využívá techniku doplnění chybějící oblasti pomocí hledání vhodně odpovídajících náhradních, neporušených políček v obrazu. Zatímco metoda difúze, jak je z názvu jasno, chybějící oblast se vyplní difuzí obrazových informací ze známé oblasti. Difuzně založené algoritmy dosáhly vynikajících výsledků pro vyplňování nestrukturovaných oblastí, mají však tendenci zavádět „hladký“ efekt ve větších chybějících oblastech.

Nejvíce proslulým programem pro tento styl úpravy je nejspíše pokročilý grafický editor, Adobe Photoshop, s nástrojem retušovacího štětce. Ovšem pokud chceme ty nejlepší výsledky, použijeme přímo nástroje vytvořené výhradně pro inpainting. Nejjednodušší metodou je vyhledávání nejpodobnějších obrazů, například z datové sady milionů obrázků, a poté chybějící části přímo vložit. Tento vyhledávací algoritmus by byl časově velmi náročný. Díky přístupu hlubokého učení a éře Big Data máme dnes k dispozici metodu, která dokáže generovat chybějící pixely v obrazu s jemnými texturami a překvapivou kvalitou.



Obrázek 12: Image Inpainting pomocí LAMA [7]

Jak vidíme na obrazu, metoda potřebuje 2 vstupy od uživatele. Obraz, který chceme rekonstruovat a masku poškozených částí. Techniku inpaintingu lze použít nejen k pouhému odstranění škrábanců, ale také k odstranění objektů, textů a dalším nechtěným částem obrazu.

### 2.1.5 Zvyšování rozlišení

V současné době tvoří upscale obrazů nedílnou součást internetu. Kvůli datovému toku bývají obrazy nahrány v horším rozlišení. V současné době už převažuje upscale právě pomocí hlubokého učení. Ta funguje tak, že k predikci obrazu s vysokým rozlišením je síť trénována na milionech obrázků s malým a vysokým rozlišením. K tomu se ještě snaží vytvořit detailey, které nikdy neexistovaly anebo byly vlivem komprese ztraceny. Aby toho bylo dosaženo, vezme matematická funkce obraz s nízkým rozlišením, který postrádá rysy a detailey, a následně jej halucinuje. Matematická funkce je známá jako model a zvětšený obraz je predikcí modelu.



Obrázek 13: Upscale pomocí Topaz Gigapixel [8]

## 2.2 Metriky používané pro měření kvality výsledné rekonstrukce

Hodnocení kvality obrazu je jedním z nejnáročnějších oborů v oblasti digitálního zpracování obrazu. Lze ho provádět subjektivně tzv. naše vlastní pozorovací schopnosti a všímání si detailů, či objektivně.

### 2.2.1 PSNR

Jedná se o nejpopulárnější metriku pro měření kvality obrazu, která ovšem nekoreluje dobře se subjektivním hodnocením. PSNR vypočítává špičkový poměr signálu a šumu v decibelech mezi dvěma obrázky. Tento poměr se používá především k určení kvality mezi původním a komprimovaným obrazem. Čím vyšší hodnota je, tím je kvalita komprimovaného nebo rekonstruovaného obrazu lepší. K definici PSNR je nutné nejdříve definovat střední kvadratickou chybu (MSE) jejíž definice zní:

Tabulka 2: Popis vzorce MSE

$\text{MSE} = \frac{\sum_{M,N} [f(m,n) - g(m,n)]^2}{M \cdot N}$	<b>M a N</b> = Počet řádků a sloupců ve vstupním obrázku. <b>m a n</b> = Indexy řádků a sloupců. <b>f a g</b> = Matice obrazů, F (referenční), G (degradován).
---	--

Vypočítanou hodnotu dosadíme do vzorce pro PSNR:

Tabulka 3: Popis vzorce PSNR

$\text{PSNR} = 20 \cdot \log_{10} \left( \frac{\text{MAX}_f}{\sqrt{\text{MSE}}} \right)$	<b>MAX<sub>f</sub></b> = Maximální hodnota signálu v referenčním obrázku.
--	---

Tento výpočet platí jen pro černobílé vstupní data, pro barevné je potřeba kompletně jiného přístupu. Jelikož je lidské oko nejcitlivější na luma informace (jas v obrázku), můžeme PSNR vypočítat převedením obrázku do barevného prostoru, který odděluje intenzitu luma kanálu, jako je například formát YCbCr. Y (luma) zde představuje vážený průměr R, G a B. G je dána největší váha, protože ji lidské oko vnímá nejsnáze. [23]

Typická hodnota pro komprimované obrázky je mezi 30 a 40 dB. Pro totožné je MSE 0, PSNR tedy není definováno.



Obrázek 14: Ukázka hodnot PSNR při kompresi [9]

Jedná se o nejvíce používanou metriku jak v oblasti strojového učení, tak použití filtrů.

### 2.2.2 SSIM

Už téměř přes dvě desetiletí hraje významnou roli při hodnocení kvality obrazu v mnoha vědních oborech. Většina objektivních technik se opírá o kvantifikaci chyb mezi referenčním a ukázkovým obrazem. Běžná metrika porovnává rozdíl v hodnotách každého odpovídajícího pixelu mezi nimi. Nicméně lidské vizuální vnímání je vysoko schopné identifikovat strukturní informace z celé scény a identifikovat rozdíly mezi informacemi z reference a vstupu. [29] Pro lepší výsledky se tedy SSIM snaží kopírovat toto chování extrahováním tří klíčových vlastností z obrazu:

**Světlonoš** – Počítá se zprůměrováním všech pixelových hodnot. Značí se písmenem  $\mu$ , kde je vzorcem:

Tabulka 4: Vzorec pro závislost světlnosti v obrazu

$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i$	$x_i$ = Hodnota pixelu $i$ na obrázku $x$ . $N$ = Celkové číslo hodnot. Porovnání světlnosti $l(x,y)$ je poté funkce $\mu_x$ a $\mu_y$
--	--

**Kontrast** – Směrodatná odchylka (druhá odmocnina rozptylu) všech hodnot pixelů. Označuje se  $\sigma$  a reprezentuje se vzorcem:

Tabulka 5: Vzorec pro závislost kontrastu v obrazu

$\sigma_x = \left( \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \right)^{\frac{1}{2}}$	Porovnání kontrastu $c(x,y)$ je poté funkce $\sigma_x$ a $\sigma_y$
--	---

**Struktura** – V podstatě rozdělíme vstupní signál s jeho směrodatnou odchylkou tak, aby výsledek měl jednotkovou směrodatnou odchylku, která umožňuje robustnější srovnání obrazů:

Tabulka 6: Vzorec pro finální strukturu

$\frac{(x - \mu_x)}{\sigma_x}$	X = Vstupní obraz.
--------------------------------	--------------------

Tento systém vypočítá index strukturální podobnosti mezi dvěma obrázky, což je hodnota mezi -1 a +1. Kde extrém +1 znamená, že zadané obrázky jsou velmi podobné až stejné, zatímco extrém -1 znamená, že jsou si velmi odlišné. [29]

Nyní máme vzorce pro určení parametrů, ale nemáme pro jejich porovnání.

**Porovnání světelnosti** – X a Y jsou parametry pro dva obrazy k měření. Funkce pro porovnání je definováno jako  $l(x,y)$ :

Tabulka 7: Vzorec pro porovnávání světelnosti

$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$	$C_1$ = Konstanta pro prevenci dělení nulou, kde vzorcem je: $C_1 = (K_1 L)^2$ L = Dynamický rozsah pixelů (standardní 8bit obrázky mají například 255)
--	---

**Porovnání struktury** –  $\mathbf{X}$  a  $\mathbf{Y}$  jsou dva porovnávané obrázky, kde  $\sigma$  označuje směrodatnou odchylku daného obrázku.

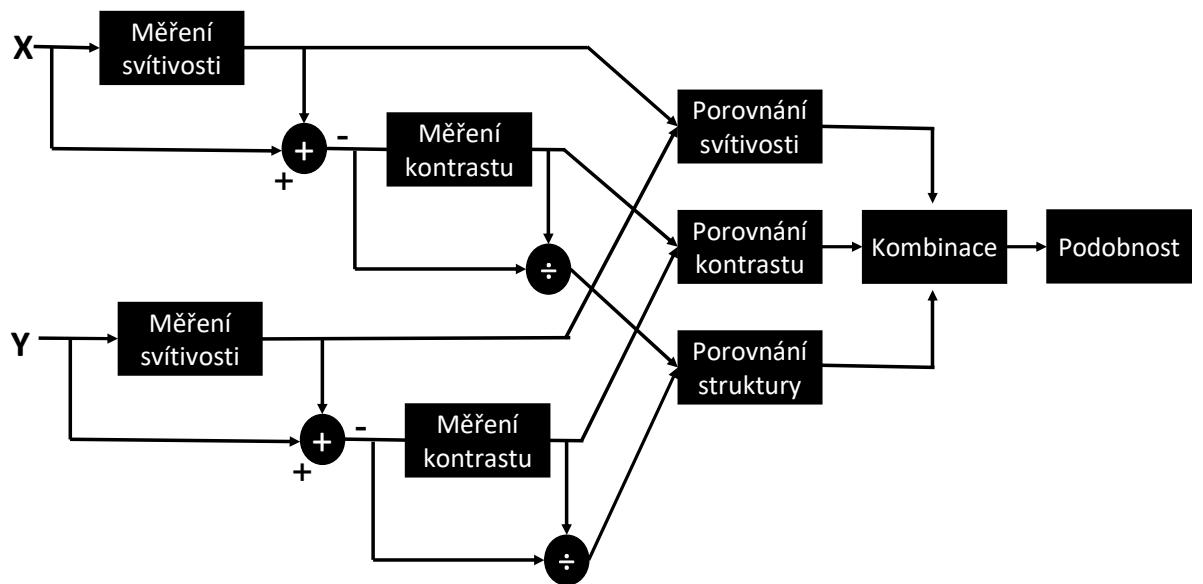
Tabulka 8: Vzorec pro porovnávání struktury

$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$	$\sigma_{xy}$ je zde definováno jako: $\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y)$
---	---

Těmito vzorcemi jsme se dostali k finálnímu základnímu tvaru vzorce funkce **SSIM**:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Kroky tohoto algoritmu jsou tedy následovně:



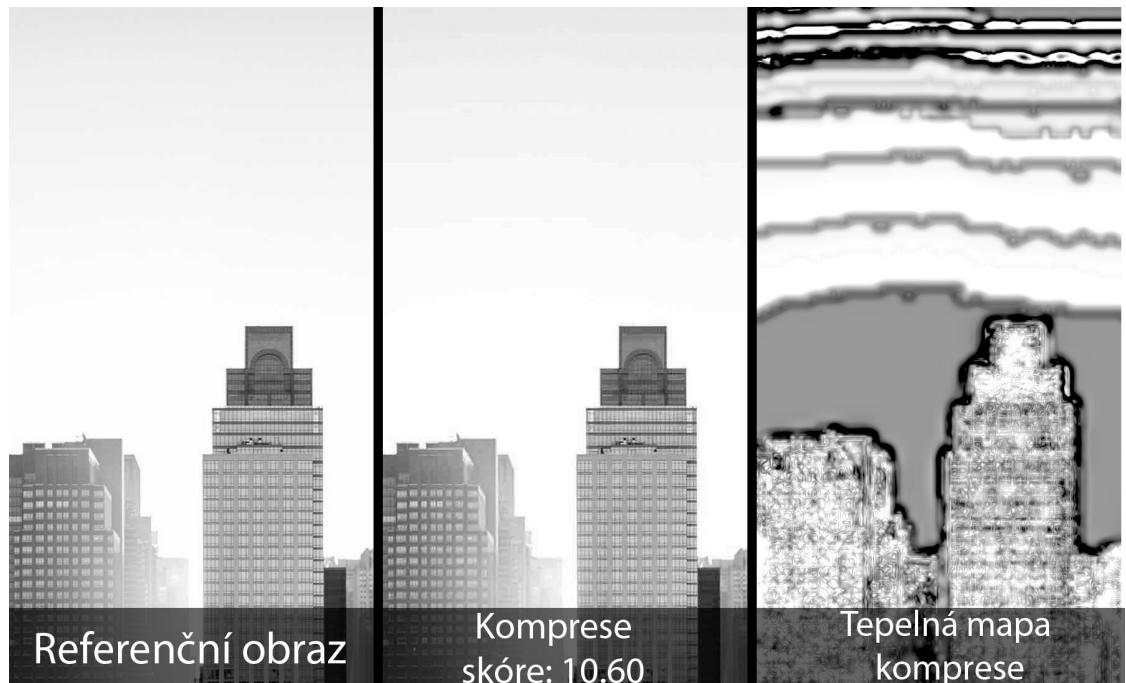
Obrázek 15: SSIM algoritmus

I když SSIM generuje užitečné výsledky, v některých případech může generovat kontraintuitivní výsledky, které mohou vést strojové učení špatným směrem. Současný výzkum grafiky a renderingu se zaměřuje především na sledování a značení paprsků Monte Carlo (pseudonáhodné) a rekonstrukci algoritmy využívající neuronové sítě. Tyto sítě často zavádějí malé odchylky během tréninku a mohly by potenciálně trpět právě kvůli nedostatkům SSIM.

### 2.2.3 Butteraugli

Metoda původně vznikla pro nástroj Guetzii od firmy Google, který má za úkol najít co velikostně nejmenší převod na JPEG, které lidské oko nedokáže odlišit od původního obrázku. Butteraugli bere v úvahu tři vizuální vlastnosti, které většina JPEG enkodérů nevyužívá. Především neaplikuje na každý RGB kanál zvlášť gama korekci. Ku příkladu, změny modré barvy v blízkosti žluté mohou být méně přesné. Dále, lidské oko má nižší prostorové rozlišení modré barvy než třeba červené a zelené, tudíž vysokofrekvenční změny v modré barvě mohou být zakódovány méně přesně a tím pádem neovlivňují výsledek měření. Posledním trikem je, že viditelnost jemných struktur v obraze záleží na množství zrakové aktivity v okolí, tudíž oblasti s velkým vizuálním šumem můžeme měřit/kódovat méně přesně. [24] Tento nástroj má i zabudovaný export do teplotní mapy, kde čím temnější odstín, tím větší rozdíl oproti referenčnímu obrazu.

Butteraugli má měřítko, které koresponduje s kvalitami JPEG úrovní enkódování. Byly vypočítány mediánem Butteraugli skóre JPEGů generovaných pomocí programu v dané kvalitě ze sady PNG obrazů. Skóre menší než 1.0 je kategorizováno jako ten nejlepší výsledek, 1.0–1.1 je přijatelné a větší než 1.1 jsou viditelné artefakty. Tato metoda je stvořena pouze pro jemné změny mezi obrazy, proto se využívá jen pro „laboratorní“ účely. Zde příklad nejhorší možné komprese na JPEG s tepelnou mapou:



Obrázek 16: Tepelná mapa komprese [10]

## 2.2.4 BRISQUE

Jedná se o nereferenční metodu, což znamená, že jedinou informaci, co algoritmus obdrží je zkreslený obraz, jehož kvalita se hodnotí a na druhém konci spektra jsou algoritmy plné reference, které jako vstup vyžadují nejen zkreslený obraz, ale také čistý referenční obraz. BRISQUE algoritmus je založen na principu, že přirozené nedotčené obrazy mají určité pravidelné statistické vlastnosti, které jsou měřitelně změněny přítomností zkreslení. Ačkoli přítomnost referenčního snímku zjednodušuje problém hodnocení kvality, praktické využití takových algoritmů jsou dosti omezené. Cílem objektivního algoritmu pro hodnocení kvality je předpovědět skóre kvality tak, aby co nejvíce korelovalo s hodnocením lidským.

BRISQUE vypočítává skóre pomocí modelu metody podpůrných vektorů trénovaných na obrazové databázi s odpovídajícími hodnotami DMOS (Průměrné rozdílově vnímané skóre). Databáze obsahuje obrázky se známým zkreslením, jako jsou kompresní artefakty, rozmazání, šum a zároveň obsahuje jejich původní verze. [25] Snímek, který má být hodnocen, musí mít alespoň jedno z těchto zkreslení, pro které byl model trénován. Jak bylo v úvodu zmíněno, čím vyšší skóre, tím horší kvalita pořízeného snímku.

Rozložení intenzity pixelů přirozených obrazů se liší od rozložení těch zkreslených. Rozdíl je poté výraznější, když normalizujeme intenzity pixelů a vypočítáme jejich rozložení nad těmi normalizovanými. Po normalizaci se intenzity pixelů řídí tvarem Bellovy křivky, zatímco u nepřirozených nikoli. Odchylka od křivky je tedy míra zkreslení obrazu.

Vzhledem k obrázku je potřeba vypočítat lokálně normalizovanou svítivost pomocí lokálního středního odčítání a následně jej vydělit lokální odchylkou, a nakonec se přidá konstanta, aby se předešlo nulovému dělení. Vzorec je tedy následovný:

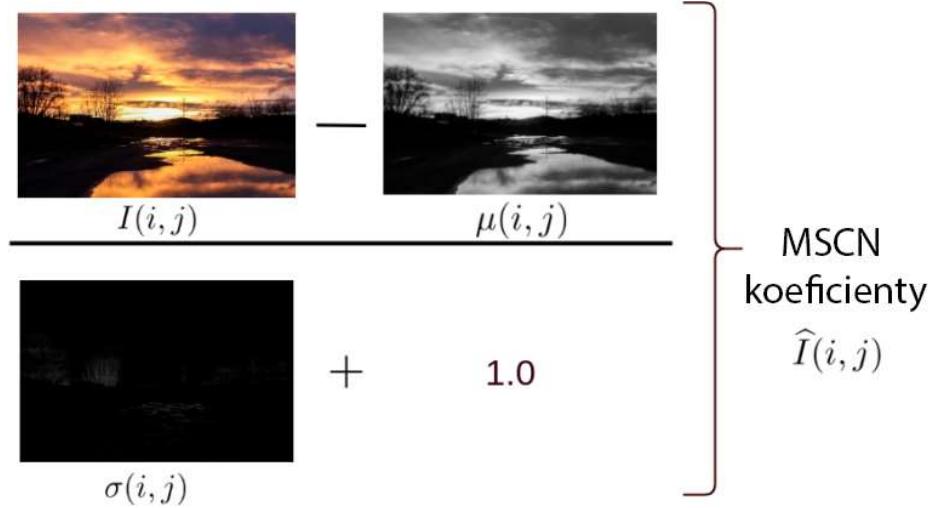
Tabulka 9: BRISQUE vzorec

$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C}$	Pokud má $I(i,j)$ rozsah $[0, 255]$ , pak $C = 1$ Pokud má $I(i,j)$ rozsah $[0, 1]$ , pak $C = \frac{1}{255}$
--	--

K vypočítání lokálně normalizované luminiscence, známé také jako průměrné odečtené kontrastní normalizované koeficienty (MSCN), musíme vypočítat lokální průměr.

Zjednodušeně řečeno, přidáme na obraz Gaussovský blur ( $\mu$ ). Nakonec vypočítáme lokální odchylku:  $\sigma(i,j) = \sqrt{\sum_{k=-K}^K \sum_{l=-L}^L w_{k,l} (I_{k,l}(i,j) - \mu(i,j))}$

Příklad vypočítání MSCN koeficientů by vypadal zhruba takto:



Obrázek 17: Vizualizace koeficientů [25]

Díky MSCN získáme dobrou normalizaci intenzity pixelů. Nicméně, rozdíl mezi přirozeným a zkresleným obrazem se neomezuje pouze na rozloučení intenzity pixelů, ale také na vztah mezi pixelem a jeho sousedy. K zachycení takových vztahů použili autoři BRISQUE metodu párového porovnání MSCN obrázku s jeho posunutou verzí. Pro nalezení párového produktu jsou čtyři orientace: Horizontální, Vertikální, Levá Diagonální a Pravá Diagonální. Z prvního obrazu jsme odvodili 5 obrazů – 1 MSCN a 4 párové obrázky produktu pro zachycení vztahů. Tyto obrazy použijeme pro výpočet vektorových rysů. [25]

Nakonec, skóre kvality vypočítáme tak, že načteme natrénovaný model a poté předpovíme pravděpodobnost pomocí podpůrných vektorů vytvořených modelem.

### 3 PRAKTIČKÁ ČÁST

#### 3.1 Různé nástroje pro rekonstrukci pomocí UI

Každý nástroj se během let vyvíjel. Některé se staly víceúčelové a některé se už dále nevyvíjely a skončili u alfa verze programu. Během této doby bylo vytvořeno spoustu softwaru, který zlepšuje výsledky, ale zároveň zvyšuje hardwarovou náročnost. Mým úkolem je najít co nejlepší a nejaktuálnější nástroje pro rekonstrukci obrazů poháněné umělou inteligencí.

##### 3.1.1 MPRNet

Jedná se kompletní balíček filtrování degradujících elementů obrazů, konkrétněji šum a rozmazání. V důsledku toho, že mnoho fotoaparátů nižších a středních tříd telefonů mají úzkou clonu a malé snímače s omezeným dynamickým rozsahem, tak často generují šum a snímky s nízkým kontrastem. Podobně jsou snímky pořízené při nevhodném osvětlení buď příliš tmavé, nebo příliš světlé.

Rekonstrukce obrazu je postup citlivý zejména na polohu, kdy je nutná korelace mezi vstupním a výstupním obrazem, zároveň je nutné zachovat všechny důležité detaily (skutečné hrany a textura obrazu) a odstranit pouze degradované části obrazu. Pro tento účel tato síť využívá víceúrovňový přístup, který zachová původní funkce obrazu vyskytující se podél hierarchie sítě, čímž se minimalizuje ztráta přesných prostorových detailů. Dokumentace popisuje, že se model nejprve učí kontextualizované funkce pomocí architektury kodér-dekodér a později je kombinuje s větví vysokého rozlišení, které zachovává všechny důležité rysy v obraze. [20] Zároveň se snaží co nejvíce zmenšit ztrátu informací. Model si totiž vyměňuje informace nejen od rané fáze do pozdní, ale zpracovává do výsledku i boční větve, ve kterých se zpracovávají rysy.

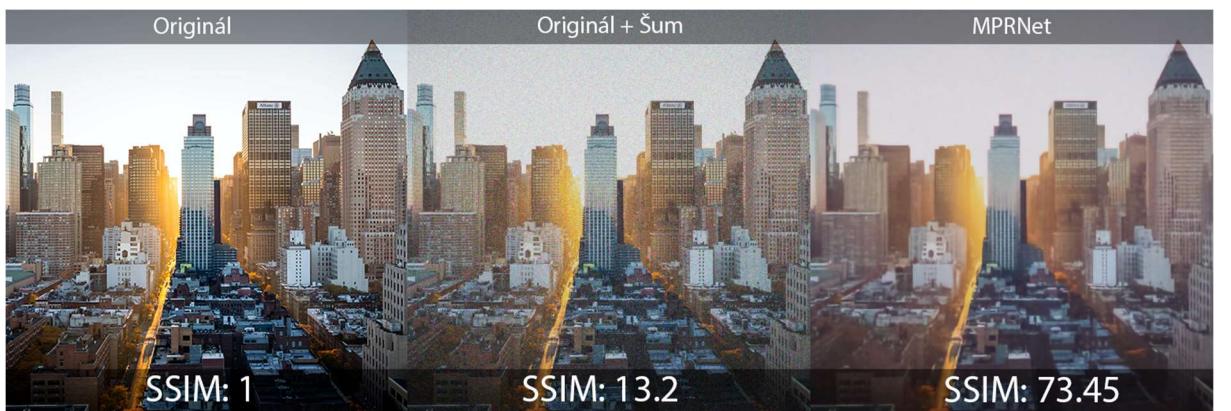
Všechny části modelu jsou založené na konvoluční neuronové síti. Ty se vyznačují použitím konvolučních a pooling vrstev, díky nim jsou schopné zpracovat takový nápor informaci, jako je například v obrazech. Konvoluční vrstva je navržená tak, že čte malé části obrazu a aplikuje na ně stejnou operaci do každé možné pozice v obrazu.

Velkou součástí je také systém SAM, který model využívá. [20] Supervised Attention Module, v překladu Modul Řízeného Dohledu, který mnohonásobně zrychluje samotný proces. Místo toho, aby na každém stupni neuronové sítě předpovídaly celý obraz a předávaly

ho na další stupeň, SAM se vloží mezi každé dva stupně a pomocí sledování vytvoří mapy pozornosti, které disponují pouze nedůležitými informacemi a do další fáze umožní šířit pouze informace užitečné.

Pro instalaci bylo využito WSL Ubuntu a virtuální prostředí pro balíčky conda. Tímto jsem docílil k oddělení veškerých nainstalovaných balíčků od zbytku systému a naprosté nezávislosti od okolí. Tato architektura využívá open-source framework PyTorch. Veškerou instalaci balíčků poté provádime dříve zmíněným balíčkovacím systémem *conda* dle návodu.

Výsledky měření ukazují, že se jedná o jeden zatím z nejvíce pokročilých nástrojů s otevřeným kódem na trhu k roku 2021.



Obrázek 18: Příklad denoisingu pomocí MPRNet [10]

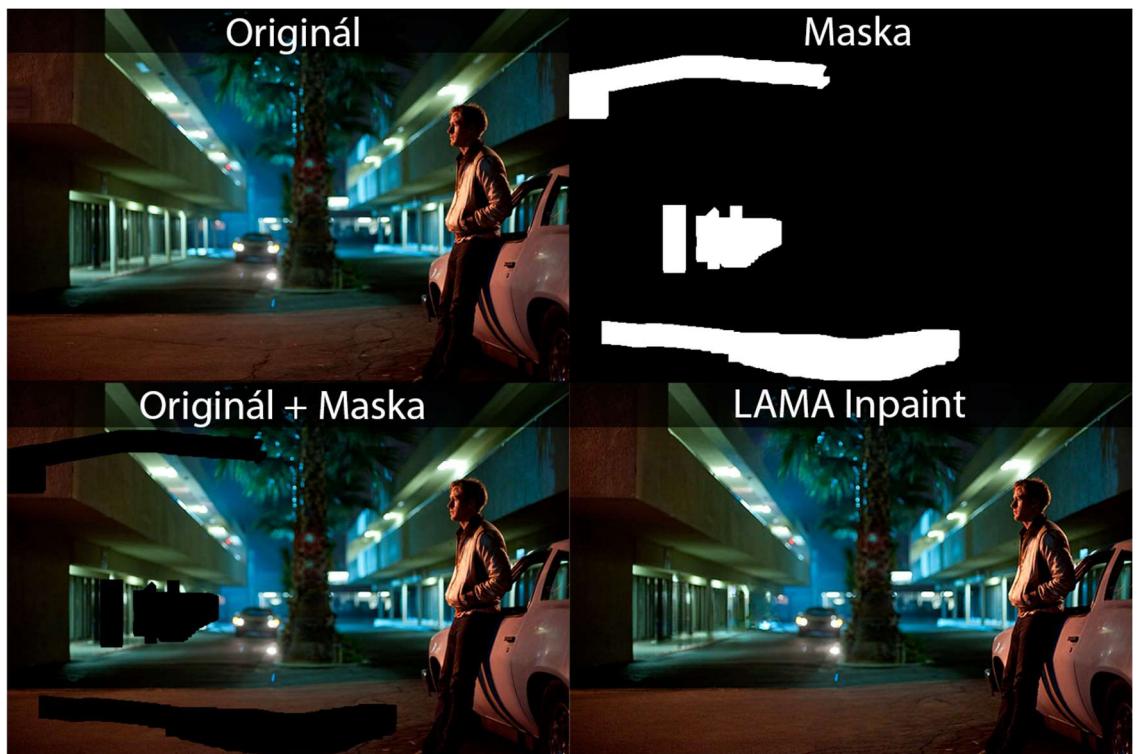
### 3.1.2 LAMA

Moderní architektury inpaintingu se i přes velký pokrok potýkají s velkými chybějícími oblastmi, které se nepodařilo vhodně doplnit. Vývojáři LAMA zjistili, že jeden z hlavních důvodů této skutečnosti je, že valná většina předešlých sítí využívá neefektivně a nedostatečně receptivní pole jak v síti pro inpainting, tak i ve ztrátové funkci. Pro zmírnění tohoto problému se využívá metoda nazvaná LArge MAsk inpainting, neboli inpainting pomocí velkých masek. Ta využívá rychlé Fourierovy konvoluce, která má receptivní pole právě v celém obraze. [22]



Obrázek 19: Typy masek v inpaintingu [22]

Vypočítání kvality inpaintingu je ze své podstaty nejednoznačné. Existuje mnoho pravděpodobných výplní pro stejně chybějící oblasti, kde kombinace stále narůstají, čím je chybějící oblast větší. Proto lze jedině vycházet z experimentů. Z nichž je jasné, že čím je chybějící oblast větší, tím je pro model těžší nahradit danou oblast z dostupných informací a tím je tedy větší ztrátová funkce. I přesto zde LAMA využívající Fourierovu konvoluci dosahuje, pro lidské oko, pozoruhodných výsledků.



Obrázek 20: Příklad inpaintingu obrazu [11]

Pro jednoduché a rychlé spuštění vývojáři poskytují sešit v Google Colab. Tato služba slouží ke spouštění a psaní python kódu skrze prohlížeč. Spadá tedy do kategorie cloud computing, jelikož kód je prováděn na vzdáleném serveru. Služba je kompletně zdarma a je nutný pouze Google účet.

### 3.1.3 SwinIR

Tento open-source software slouží k vícenásobnému zvětšení rozlišení obrazu s co nejmenší ztrátou informací. Zatímco nejmodernější metody obnovy obrazu jsou založené na konvolučních neuronových sítích, zde se používá model Tranformer, který dosahuje pozoruhodných výsledků.



Obrázek 21: Upscale pomocí SwinIR [12]

Transformer je model hlubokého učení, který využívá mechanismus self-attention (sebe pozornosti) a diferencovaně váží významnost každé části vstupních dat. Na rozdíl od podobných sítí, transformátory nemusí nutně zpracovávat data v pořadí. Self-attention mechanismus spíše poskytuje kontext pro libovolnou pozici ve vstupní sekvenci. Ovšem tento přístup má dvě nevýhody. Za prvé, model obvykle rozděluje vstupní obraz na políčka s pevnou velikostí, což například znamená, že okrajové pixely nemohou pro obnovu obrazu využít sousední pixely, které jsou mimo dané políčko. Za druhé, obnovený obraz může zanechávat hraniční artefakty. [26] To lze vyřešit překrýváním políček, to by však znamenalo výpočetní zátěž. Oba problémy vyřešíme použitím Swinova transformátoru, který kombinuje výhody a mechanismy Konvolučních sítí a Transformátoru.

SwinIR se skládá ze tří podstatných modulů. Povrchové funkce extrakce, hluboká funkce extrakce a Vysoko kvalitní rekonstrukční modul. [26] Povrchová funkce extrahuje povrchové rysy obrazu a k tomu využívá konvoluční vrstvu. Touto funkcí tedy zachováme

nízkofrekvenční informace v obrazu. Hluboká funkce extrahuje hloubkové rysy, kde každý blok využívá několik vrstev Swinova transformátoru pro interakci mezi okny. Nakonec jsou povrchové a hloubkové rysy sloučeny právě rekonstrukčním modulem.

Oproti ostatním metodám na bázi CNN je výsledek až o půl dB lepší a k tomu se i počet potřebných parametrů při trénování sítě zmenšil až o 67%. [26] Pro představu, síť RCAN, postavená čistě na CNN potřebuje pro slušný výsledek 15.6 milionů parametrů, SwinIR si vystačí pouze s 11.9 milionu. Pro použití je znovu připraven Google Colab sešit a znovu je využit framework PyTorch.

### 3.1.4 BOPBTL

Jedná se o oficiální Microsoft nástroj z odvětví výzkumu Machine Learningu, který slouží k obnově starých fotek neboli „oživení starých fotografií“. Na rozdíl od běžných úloh obnovy obrazu je obnova starých fotografií náročnější. Staré fotografie totiž obsahují mnohem složitější degradaci, kterou je obtížné realisticky upravit. Síť se proto obvykle nemůže dobře zobecnit pouhým učením ze syntetických dat. Navíc, staré fotografie jsou s velkou pravděpodobností složeny z více degradací, a proto vyžadují různé strategie obnovy. Nestrukturované vady, ku příkladu šum, rozmažání či vyblednutí barev, lze vyřešit pomocí filtrů a strukturované vady jako například škrábance a skvrny by měly být inpaintovány. Ovšem generalizace obou metod je o mnoho těžší.



Obrázek 22: Ukázkový příklad využití nástroje [27]

Klíčový je zde překlad obrazu do tří domén. Skutečná fotografie, syntetická doména, kde obrazy trpí umělou degradací a odpovídající výsledek. Tato třetice využívá neoznačené reálné fotografie a velké množství syntetických dat spojené právě s výsledkem. [27] Přímé mapování reálných fotografií na čisté snímky je příliš obtížné, protože nejsou spárované. Tedy nelze

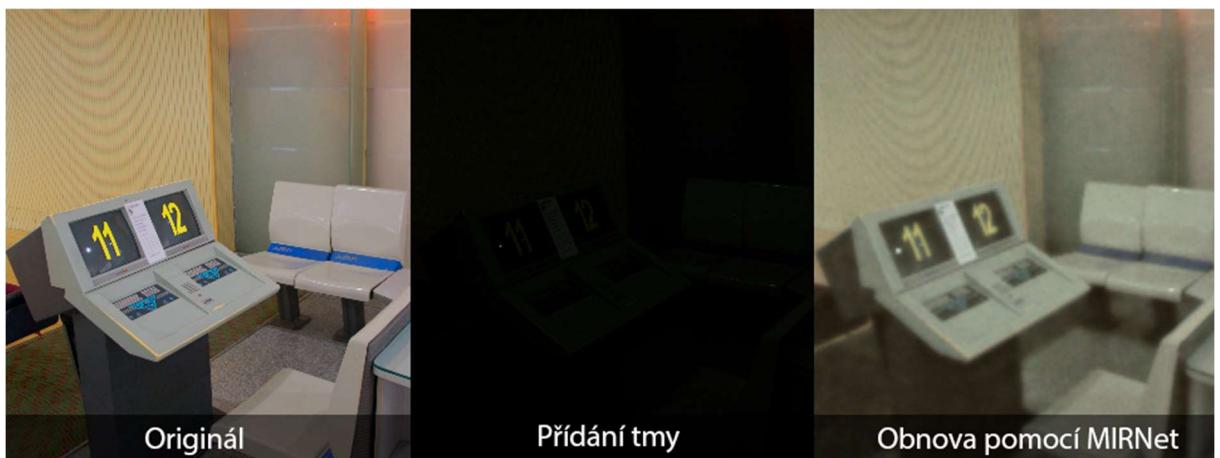
využít učení pod dohledem, proto se učení rozdělilo do dvou fází. Nejprve trojici přeložíme do odpovídajících latentních prostorů a poté jej naučíme obnovu obrazu v latentním prostoru. Naučíme tedy síť převod latentního prostoru poškozených dat do skutečné formy. [27]

Tato obnova se soustředí pouze na lokální rysy. Nicméně obnova strukturovaných defektů vyžaduje věrohodný inpainting, který musí brát v úvahu závislosti v dlouhém dosahu. Proto do sítě musíme dodat externí nelokální blok. Jelikož u většiny poškozených fotografií nemáme originál, je zde vhodné pro měření využít BRISQUE metodu.

Microsoft k tomuto nástroji poskytl celý kód a dokonce dodal jednoduché GUI pro ovládání. I tak má nástroj sešit v Google Colab. Pro obnovu starých fotografií je právě tento nástroj špička na trhu, jelikož je stále průběžně aktualizován.

### 3.1.5 MIRNet-TFJS

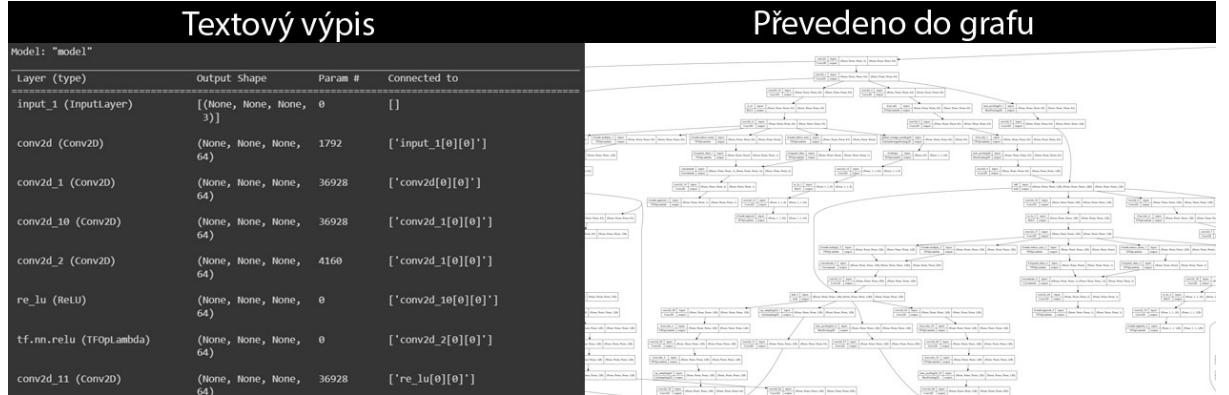
Jedná se o nástroj specializovaný na vysoko kvalitní zesvětlení velmi tmavých fotek. Podobný nástroj například používá Samsung, ve své nové generaci, jako „Night Mode“. St MIRNet představuje novou architekturu přístupu k tomuto problému, jejímž cílem je udržovat reprezentace s vysokým rozlišením v celé síti a získávat kontextové informace z reprezentace s nízkým rozlišením.



Obrázek 23: Zesvětlení pomocí MIRNet [28]

Jádrem celé architektury je multiškálový reziduální blok obsahující paralelní konvoluční proudy s více rozlišeními pro extrakci rysů ve více měřítcích, následná výměna mezi proudy, mechanismy prostorové a kanálové všímvosti pro zachycení kontextových informací a jejich následná agregace. [28]

Při zkoušení softwaru nám model umožňuje nahlédnout do jeho funkcionality souhrnem. V něm například vidíme, co vše model používá za informace a dovoluje nám zobrazit: název a typ všech vrstev v modelu, výstupní tvar pro každou vrstvu, počet váhových parametrů každé vrstvy, vstupy, které každá vrstva přijímá a celkový počet trénovatelných a netrénovatelných parametrů modelu.



Obrázek 24: Parametry modelu

Výsledek tohoto souhrnu poté program umožňuje uložit do grafu. Ze souhrnu se dozvídáme, že celkově bylo potřeba přes 36 milionů parametrů.

Model je postaven na frameworku TensorFlow. Jedná se o bezplatnou softwarovou knihovnu s otevřeným zdrojovým kódem, která se zaměřuje především na trénink hlubokých sítí. Byla vyvinuta týmem Google Brain pro interní použití. Znovu lze využít přednastavený Google Colab sešit, ovšem poskytují se zde i jiné způsoby implementace.

### 3.2 Program pro zjištění kvality rekonstruovaných obrazů

Kvalita obrazu se může vztahovat k úrovni rekonstrukce, komprimace a dalších metod.

Jelikož jsou subjektivní metody nákladné, protože vyžadují velký počet lidí a nelze jej automatizovat v reálném čase, cílem je tedy navrhnout algoritmy, které by byly v souladu s lidským hodnocením

#### 3.2.1 Důležité faktory hodnocení

Proces hodnocení je ovlivněn několika faktory. Mezi nejhlavnější část obrazu patří převážně **ostrost**. Ta je buďto ovlivněna objektivem nebo kvalitou komprimace obrazu. Jedná se o vlastnost převážně žádanou, jelikož po většinu času koreluje s kvalitou. Naopak **šum** je faktor nežádaný. Jedná se o náhodné kolísání hustoty obrazu, které se v obraze projevuje jako

zrnění. Během zpracovávání obrazu se můžeme setkat s **artefakty**. Důvodem výskytu bývá nedostatečně naučená neuronová síť.

V teoretické části jsme si vysvětlili jednotlivé metody používané právě pro měření a zde je využijeme přímo v kódu.

### 3.2.2 Kód

Pro vytvoření jsem si vybral jazyk Python, který se vyznačuje svojí jednoduchostí a přehledností. Program má tři hlavní části. Import knihoven, komentáře a kód s definicemi.

**Část importu** popisuje, které všechny doplňky program používá. Zde vidíme převážně matematické doplňky jako například *math*, *cv2* a *numpy*. Následovně pro předání cesty souboru přes argument programu využíváme modul *argparse*. Ten jej poté vkládá do pole a lze s ním jednodušeji pracovat.

```
import math
import cv2
import numpy as np
import argparse
from skimage.metrics import structural_similarity
from BRISQUE_MODULES import *
import libsvm.svmutil as svmutil
from svm import *
from svmsvmutil import *
```

Obrázek 25: Hlavička programu

Jelikož BRISQUE metrika je složitější než ostatní metriky, rozhodl jsem se program rozdělit na dva soubory, kde do hlavní části vložíme pouhou referenci na soubor s funkcemi, které BRISQUE používá. Soubor nazván BRISQUE\_MODULES obsahuje nutné definice funkcí pro správnou funkci programu.

Syntax pro úspěšné zavolání programu je následovná. Jelikož se jedná o python program, musíme jej interpretovat. Tím docílíme napsáním *python* a soubor, který chceme přeložit. Do programu jsem vložil dva povinné argumenty. Argument *-o* jako *original* a *-c* jako *contrast*.

```

Kód
ap = argparse.ArgumentParser()
ap.add_argument("-o", "--original", required=True)
ap.add_argument("-c", "--contrast", required=True)
args = vars(ap.parse_args())

```

Terminál

```

python IMAGE_MEASUREMENT.py -o BLADER.png -c BLADER_COMPRE.jpg

```

Obrázek 26: Definice a použití argumentů programu

Hodnotu v argumentu předáme a následně převedeme do odstínů šedé barvy pro snadnější zpracování.

```

# čtení obrazu z argumentů
imageA = cv2.imread(args["original"])
imageB = cv2.imread(args["contrast"])
# převedení do odstínů šedi
grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)

```

Obrázek 27: Zpracování obrazu

Následně definujeme první dvě metody, které budeme v programu využívat, tedy PSNR a SSIM. Jak vidíme, definice šla hladce dle vzorů definovaných v teoretické části práce. Ovšem BRISQUE byl o mnoho složitější. Nejedná se totiž o jednoduchou matematickou funkci.

```

def PSNR(img1, img2):
    #
    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return 100
    #
    PIXEL_MAX = 255.0
    #
    PSNRScore = 20 * math.log10(PIXEL_MAX / math.sqrt(mse))
    return PSNRScore

#####
##### SSIM(img1, img2):
    (SSIMScore, diff) = structural_similarity(img1, img2, full=True)
    diff = (diff * 255).astype("uint8")
    return SSIMScore

#####

```

Obrázek 28: Definice funkcí PSNR a SSIM

BRISQUE potřebuje oproti minulým metodám o mnoho externích informací navíc. Jak víme, metoda počítá výsledné skóre pomocí modelu regrese s podpůrnými vektory, vycvičených na databázi obrázků s odpovídajícími hodnotami názorového skóre. Měřený obrázek tedy musí mít alespoň jedno ze zkreslení, na které byl trénován.

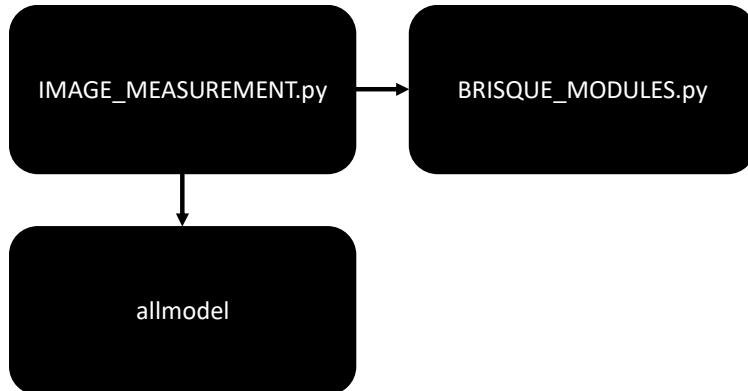
```

1 svm_type epsilon_svr
2 kernel_type rbf
3 gamma 0.05
4 nr_class 2
5 total_sv 770
6 rho -155.845
7 probA 6.34795
8 SV
9 -1024 1:-0.597198 2:-0.143425 3:-0.250373 4:0.434151 5:-0.770629 6:-0.120473 7:-0.261166 8:0.437833 9:-0.749714
10:-0.204235 11:-0.170752 12:-0.101459 13:-0.597951 14:-0.584089 15:-0.168594 16:-0.110996 17:-0.594929 18:-0.593067
19:0.41183 20:0.210422 21:0.395286 22:0.165987 23:-0.339455 24:0.391707 25:0.373055 26:0.0738007 27:-0.299791
28:0.239653 29:0.50318 30:-0.356149 31:-0.0408499 32:-0.288528 33:0.533753 34:-0.386524 35:-0.0501816 36:-0.314717
10 -1024 1:-0.750162 2:-0.467213 3:-0.516279 4:0.449112 5:-0.921984 6:-0.450775 7:-0.528675 8:0.11969 9:-0.864933
10:-0.64794 11:-0.434466 12:0.150394 13:-0.846166 14:-0.791966 15:-0.432509 16:0.129298 17:-0.843114 18:-0.800605
19:-0.0421003 20:-0.126157 21:-0.0164766 22:0.436398 23:-0.675814 24:0.0658424 25:0.0178726 26:-0.200528 27:-0.481642
28:-0.395445 29:0.118467 30:-0.0683621 31:-0.465541 32:-0.551679 33:0.141398 34:-0.234465 35:-0.431874 36:-0.600774
11 1024 1:-0.754095 2:-0.57969 3:-0.545981 4:0.386669 5:-0.95345 6:-0.599779 7:-0.561229 8:0.148354 9:-0.909084
10:-0.719219 11:-0.426844 12:0.221294 13:-0.90574 14:-0.851418 15:-0.424881 16:0.263192 17:-0.90921 18:-0.848744
19:0.0128477 20:-0.273723 21:-0.0329473 22:0.412618 23:-0.758295 24:-0.151472 25:-0.110853 26:0.177895 27:-0.687629
28:-0.364115 29:0.141398 30:-0.229309 31:-0.580118 32:-0.730525 33:0.133754 34:-0.108961 35:-0.608392 36:-0.707541
12 1024 1:-0.827579 2:-0.730515 3:-0.684592 4:0.30859 5:-0.987043 6:-0.754401 7:-0.657476 8:0.365962 9:-0.985231
10:-0.802138 11:-0.518305 12:0.473747 13:-0.97618 14:-0.902786 15:-0.520988 16:0.49167 17:-0.976604 18:-0.902236

```

Obrázek 29: Vygenerované BRISQUE vektory

Z toho tedy vyplývá, že námi sestrojený program potřebuje tři nutné soubory ke správné funkcionalitě. A to hlavní část programu *IMAGE\_MEASUREMENT.py*, část nutnou pro import podpůrných funkcí *BRISQUE\_MODULES.py* a nakonec vygenerované vektory, *allmodel*. Hierarchie programu je tedy jednoduchá.



Obrázek 30: Hierarchie programu

Konec programu obsahuje výpis naměřených hodnot obrazu. Vše je zaokrouhleno na dvě desetinná místa pro lepší čitelnost. Pro přehlednost jsem do programu přidal průměr naměřených hodnot, konkrétněji průměr SSIM a BRISQUE.

PSNR do průměru nebylo zahrnuto z důvodu jiné míry měření. Jak víme z teoretické části, SSIM měří od 0.1 – 1. Z tohoto důvodu jsem se rozhodl výslednou hodnotu násobit 100 pro lepší přehlednost.

```
brisqueOut = round(BRISQUE(imageA), 2)
ssimOut = round(SSIM(grayA,grayB)*100, 2)
psnrOut = round(PSNR(imageA,imageB), 2)
qualList = [brisqueOut,ssimOut]
avgQual = round(sum(qualList)/len(qualList),2)

print("-----")
print("BRISQUE KVALITA: {}".format(brisqueOut))
print("SSIM KVALITA: {}".format(ssimOut))
print("PSNR KVALITA: {}".format(psnrOut))
print("-----SOUHRN-----")
print("ZPRŮMĚROVANÁ KVALITA (BRISQUE, SSIM): {}".format(avgQual))
print("-----")
```

Obrázek 31: Výpis programu

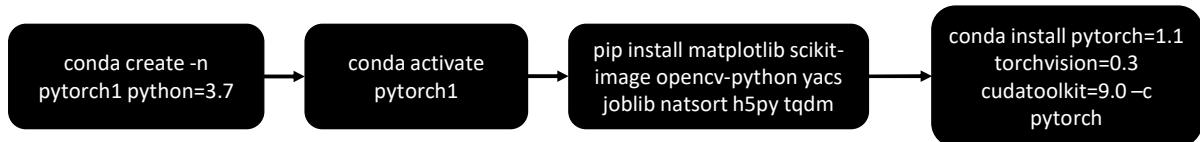
### 3.3 Implementace algoritmu a případné vylepšení

K implementaci je dostupných mnoho programů, ovšem jejich funkčnost a spolupráce s námi vytvořeném prostředí pokulhává. Rozhodl jsem se implementovat MPRNet. Síť, která slouží především k deblurringu a denoisingu obrazu a k tomu všemu dosahuje obdivuhodných výsledků.

#### 3.3.1 Instalace

Jedná se o tu nejtěžší část, jelikož tento typ softwaru není určen pro širokou veřejnost. Jeho instalace vyžaduje určitou úroveň znalostí jazyku Python. Z důvodů doporučení využití systému Ubuntu jsem se rozhodl nainstalovat WSL Ubuntu pro Windows. Jedná se o subsystém Linuxu, který umožňuje nativní běh linuxových spustitelných souborů v prostředí Windows. Dále namísto balíčkovacího systému pip doporučují *conda*.

Pro zpřehlednění si vytvoříme virtuální prostředí pomocí příkazu `conda create -n pytorch1 python=3.7`, kde `pytorch1` je název virtuálního prostředí a `python=3.7` je verze interpretu, která se bude využívat. A ke konci nainstalujeme všechny důležité balíčky, které je nutno importovat pro správný chod programu. Výsledná posloupnost příkazů by vypadala takto:



Obrázek 32: Postup instalace

### 3.3.2 Hierarchie složky

Pro správnou funkcionality programu je vyžadováno spoustu souborů a složek, bez kterých by se program neobešel.

Deblurring	18.04.2022 15:05	Složka souborů
Denoising	18.04.2022 15:05	Složka souborů
pytorch-gradual-warmup-lr	18.04.2022 15:05	Složka souborů
.gitignore	18.04.2022 15:05	Git Ignore Source ...
demo.py	18.04.2022 15:05	Python Source File
LICENSE.md	18.04.2022 15:05	Markdown Source ...
README.md	18.04.2022 15:05	Markdown Source ...

Obrázek 33: Struktura složek

Každá složka a soubor má svůj podíl na tom, jak program funguje. Znalost, co k čemu patří může zásadně ovlivnit efektivitu práce. Na první pohled zde vidíme nedůležité soubory jako je například README, různé git konfigurace a `pytorch-warmup`, který slouží pro trénink sítě. Poté zde vidíme dvě hlavní složky, *Deblurring* a *Denoising*. V nichž se nachází především před-natřenované modely, které nám přijdou vhod. Bez nich bychom si museli natřenovat model sami, a k tomu je nutný lepší hardware. Model jako samotný používá PTH soubor, který má následující podobu:

1	8002	8a0a	6cfc	9c46	f920	6aa8	5019	2e80
2	024d	e903	2e80	027d	7100	2858	1000	0000
3	7072	6f74	6f63	6f6c	5f76	6572	7369	6f6e
4	7101	4de9	0358	0d00	0000	6c69	7474	6c65
5	5f65	6e64	6961	6e71	0288	580a	0000	0074
6	7970	655f	7369	7a65	7371	037d	7104	2858
7	0500	0000	7368	6f72	7471	054b	0258	0300
8	0000	696e	7471	064b	0458	0400	0000	6c6f
9	6e67	7107	4b04	7575	2e80	027d	7100	580a
10	0000	0073	7461	7465	5f64	6963	7471	0163
11	636f	6c6c	6563	7469	6f6e	730a	4f72	6465
12	7265	6444	6963	740a	7102	2952	7103	2858
13	1600	0000	7368	616c	6c6f	775f	6665	6174
14	332e	302e	7765	6967	6874	7104	6374	6f72
15	6368	2e5f	7574	696c	730a	5f72	6562	7569
16	6c64	5f74	656e	736f	725f	7632	0a71	0528
17	2858	0700	0000	7374	6f72	6167	6571	0663

Obrázek 34: PTH soubor

Jak vidíme, uvnitř souboru se jedná prakticky jen o vygenerované hodnoty při natrénovaní čili binární soubor. PTH soubory jsou typické pouze pro PyTorch.

Jako hlavní soubor je tu *demo.py*. Jedná se, laicky řečeno, o takovou „spojku“ mezi architekturou a modelem. Uživatel prakticky využívá jen tento soubor, zbytek ho nemusí zajímat, jelikož se jedná o soubory nutné k natrénovaní sítě. Výsledky úpravy obrazu se poté ukládají do námi zvolené složky.

Zbytek složek, například v Deblurringu, jsou buďto příklady použití nebo různé utility, které nám usnadňují práci. Tím pádem nám z tohoto vyplývá, že my jako uživatelé se budeme starat pouze o hlavní python soubor a o modely. Také jsou zde skripty určené pro trénování naší vlastní sítě a různé statistiky.

Datasets	18.04.2022 15:05	Složka souborů
pretrained_models	19.04.2022 16:36	Složka souborů
utils	18.04.2022 15:05	Složka souborů
config.py	18.04.2022 15:05	Python Source File
data_RGB.py	18.04.2022 15:05	Python Source File
dataset_RGB.py	18.04.2022 15:05	Python Source File
evaluate_GOPRO_HIDE.m	18.04.2022 15:05	Objective C Sourc...
evaluate_RealBlur.py	18.04.2022 15:05	Python Source File
losses.py	18.04.2022 15:05	Python Source File
MPRNet.py	18.04.2022 15:05	Python Source File
README.md	18.04.2022 15:05	Markdown Source ...
test.py	18.04.2022 15:05	Python Source File
train.py	18.04.2022 15:05	Python Source File
training.yml	18.04.2022 15:05	Yaml Source File

Obrázek 35: Deblurring složka

### 3.3.3 Spuštění programu

Po instalaci lze snadno inicializovat program se syntaxí: `python demo.py --task Název_úkolu --input_dir cesta_k_souborům --result_dir cesta_k_výsledku`. Program nabízí celkem tři různé možnosti pro opravu obrazu. Odstranění šumu, rozmazání a dešťových artefaktů. A právě typ úkolu se mění podle switche `--task`, kde jsou dostupné možnosti pojmenované jako: Deblurring, Denoising a Deraining. V mé případě bylo nutné obrazy downscalovat ideálně na rozlišení 500x500. I přes 6 GB VRAM u větších snímků dochází k Runtime chybě.

```
RuntimeError: CUDA out of memory. Tried to allocate 440.00 MiB (GPU 0; 6.00 GiB total capacity; 4.68 GiB already allocated; 0 bytes free; 419.09 MiB cached)
```

Obrázek 36: Runtime CUDA error

### 3.3.4 Vylepšení

Zde jsem si položil otázku. Dopadl by výsledek lépe, kdybychom jej následně dali upscalovat? Odpověď bohužel není tak jednoznačná. I přesto, že downscale přináší určitou úroveň degradace, může naopak upscale degradaci ještě zvýšit. Pro upscale jsem využil již dříve popsané SwinIR. To podle dřívějších testů fungovalo skvěle.

Souborovou strukturu má podobnou jako MPRNet, je tu hlavní soubor *main\_test\_swinir.py*, který znova funguje jako spojka mezi modelem a architekturou. Mým cílem bylo obě síťe spojit do sebe a zautomatizovat. Jednoduše řečeno, MPRNet by provedl Deblur/Denoise a výsledek předal SwinIR. Mým nápadem bylo, vytvořit kompletní nový python skript, třeba *Main.py*, který by zavolal oba hlavní soubory sítí. Základ je jasný, potřebujeme implementovat funkce z obou sítí a následně přidat nutné argumenty.

```
import argparse
from MPRNet import *
from SwinIR import *
#from natsort import natsorted

parser = argparse.ArgumentParser(description='Demo MPRNet')
parser.add_argument('--input', type=str, help='Input images')
parser.add_argument('--result', type=str, help='Directory for results')
parser.add_argument('--task', required=True, type=str, help='Task to run', choices=['Deblurring', 'Denoising'])

args = parser.parse_args()
```

Obrázek 37: Kód Main.py

Jako první do vložíme hlavní funkci MPRNet. Nápad byl takový, že v *Main.py* bude hlavní program MPRNetu a dodatečné funkce budou v původním souboru. Tím by se docílilo určité přehlednosti v programech a lze by je šlo snadno rozlišit. MPRNet byl úspěšně implementován a bylo na čase přejít na SwinIR. Byl jsem si vědom, že ne vždy uživatel chce provádět upscale. Proto jsem do programu přidal rozhodovací část, kde při stisknutí „y“ se upscale provede a při stisknutí „n“ se ukončí. Tím jsem docílil využitím rozhodovací větve:

```
answer = None
while answer not in ("y", "n"):
    answer = input("Upscale? [y/n]: ")
    if answer == "y":
        os.system("cmd.exe /C SwinIR.py --task real_sr --scale 4")
    elif answer == "n":
        print("Exiting program...")
    else:
        print("Y/N")
```

Obrázek 38: Rozhodovací větev v Main.py

Vše do této části fungovalo tak jak má, ovšem narazilo se na problém. Jelikož je MPRNet sestrojený v condě ve WSL Ubuntu, je nutné stejně sestrojit SwinIR podobně. Proto jsem vytvořil nové virtuální prostředí. Původní nápad byl, že by se programově přepnulo mezi prostředími. Bohužel SwinIR využívá verzi modulu cv2, který potřebuje *libx264.so.138*, jež v současné době už neexistuje a jeho instalace není možná. Sami vývojáři tvrdí, že nedoporučují Linux WSL, jelikož neposkytuje úplné API prostředí.

```
Traceback (most recent call last):
  File "main_test_swinir.py", line 2, in <module>
    import cv2
ImportError: libx264.so.138: cannot open shared object file: No such file or directory
```

Obrázek 39: Chyba při implementaci SwinIR

Chtěl jsem se vyhnout tomu, aby program byl „crossplatform“. Bohužel, nešlo to jinak, jelikož fungoval pouze v prostředí ubuntu a Python 3.7 verzí. S tímto problémem padá možnost zautomatizování, rozhodl jsem se tedy aspoň vyzkoušet, zda po upscaleu nabývá lepších hodnot.



Obrázek 40: Výsledný obraz [13]

Bohužel se dozvídáme, že SwinIR nedokáže efektivně zpracovávat obličeje. Jelikož převážně obličeje a celý obraz se zdá být velice „umělý“. Dosahuje to až efektu „uncanny valley“, kdy upscalovaný obličeje vyvolává znepokojivé pocity a downscalovaná verze se zdá být prozatím pro vjem člověka lepší.

## **4 ZÁVĚR**

I přes neuspokojivé vylepšení si myslím, že má práce dopadla úspěšně. První síť jsem plně implementoval a je kompletně funkční. K tomu všemu, kompletní balíček měřících technik funguje bez problémů a při psaní práce mi byl vhodným pomocníkem. Věřím, že tomuto tématu se nadále budu věnovat i mimoškolně, jelikož mě psaní práce bavilo. Směr, kterým se image processing vyvíjí má dle mého nepředstavitelnou budoucnost, kterou je potřeba uchopit. Jsem vděčný za možnost poskytnutí psaní tohoto tématu, jelikož jsem se toho dozvěděl opravdu mnoho v odvětví, které mě zajímalo již od třetího ročníku, jelikož volně navazují na svoji minulou ročníkovou práci.

## POUŽITÁ LITERATURA

- [1] Grayscale [online]. In: . 2006 [cit. 2022-04-20]. Dostupné z: <https://www.science.widener.edu/observatory/031506/M51all.jpg>
- [2] SVOBODA, Pavel, Michal HRADIŠ, Lukáš MARŠÍK a Pavel ZEMČÍK. License plates from a surveillance system and theirs reconstructions using direct CNN deblurring. [online]. In: . [cit. 2022-04-20]. Dostupné z: <https://www.arxiv-vanity.com/papers/1602.07873/>
- [3] Last Night In Soho [online]. In: . [cit. 2022-04-20]. Dostupné z: [https://media1.popsugar-assets.com/files/thumbor/nEb6YOstQrYIY5wdhbdsZHNWME/fit-in/2048xorig/filters=format\\_auto-!!-:strip\\_icc-!!-/2021/05/25/910/n/1922283/tmp\\_ndwtNV\\_9e083423f329ce41\\_last-night-in-soho-4139\\_D056\\_00238\\_R\\_rgb.jpg](https://media1.popsugar-assets.com/files/thumbor/nEb6YOstQrYIY5wdhbdsZHNWME/fit-in/2048xorig/filters=format_auto-!!-:strip_icc-!!-/2021/05/25/910/n/1922283/tmp_ndwtNV_9e083423f329ce41_last-night-in-soho-4139_D056_00238_R_rgb.jpg)
- [4] GREY, Alex. Lateralus. In: Alex Grey [online]. 2001 [cit. 2022-04-20]. Dostupné z: <https://www.alexbrey.com/art-images/tool-dissectional-alex-grey-watermarked.jpg>
- [5] Long Live The Smiths' 'Complete Works'. In: [online]. 2001, 26. 11. 2011 [cit. 2022-04-20]. Dostupné z: [https://media.npr.org/assets/img/2012/01/23/the\\_smiths\\_complete\\_lp\\_cover\\_wide-a0f56c58a1e5bfef7055d9b42e17cb7e45d1865e.jpg?s=1400](https://media.npr.org/assets/img/2012/01/23/the_smiths_complete_lp_cover_wide-a0f56c58a1e5bfef7055d9b42e17cb7e45d1865e.jpg?s=1400)
- [6] Last Night In Soho Wallpaper. In: Imdb [online]. [cit. 2022-04-20]. Dostupné z: [https://m.media-amazon.com/images/M/MV5BY2Y0YTExZmQtNDhjZC00ZGY2LWI2ZTItdNDM2MTcwN2YwZTAyXkEyXkFqcGdeQXZ3ZXNsZXk@.\\_V1\\_.jpg](https://m.media-amazon.com/images/M/MV5BY2Y0YTExZmQtNDhjZC00ZGY2LWI2ZTItdNDM2MTcwN2YwZTAyXkEyXkFqcGdeQXZ3ZXNsZXk@._V1_.jpg)
- [7] Taxi Driver (1976) New York Neo Noir Masterpiece. In: Noirsville [online]. [cit. 2022-04-20]. Dostupné z: [https://4.bp.blogspot.com/-UK4MmkYY-sk/V8UBHwl9e9I/AAAAAAAFAPI/1B0\\_eZs5eCUTGdCQMX2mPe1rPmTr247GgCLcB/s640/Noirish%2BTaxi%2BDriver%2B%25281976%2529.jpg](https://4.bp.blogspot.com/-UK4MmkYY-sk/V8UBHwl9e9I/AAAAAAAFAPI/1B0_eZs5eCUTGdCQMX2mPe1rPmTr247GgCLcB/s640/Noirish%2BTaxi%2BDriver%2B%25281976%2529.jpg)
- [8] CARSON, David. Fotografiks [online]. In: . 1999 [cit. 2022-04-20]. Dostupné z: [https://live.staticflickr.com/4013/4493730773\\_376788985d\\_b.jpg](https://live.staticflickr.com/4013/4493730773_376788985d_b.jpg)

- [9] Blade Runner 2049 [online]. In: [cit. 2022-04-20]. Dostupné z: <https://images.squarespace-cdn.com/content/v1/58e41216414fb56455ed6f0e/1507884671151-TIAW67LA0LJTIQP7X7NW/Blade+Runner+2049+2.jpg?format=1500w>
- [10] New York City [online]. In: 2017 [cit. 2022-04-20]. Dostupné z: <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ4SP-UF6fujY3CgqgYAlmi81buD46htbmXSQ&usqp=CAU>
- [11] Drive [online]. In: [cit. 2022-04-20]. Dostupné z: <https://craigkempsdsblog.wordpress.com/2014/12/05/visual-styles-and-film-analysing/>
- [12] ALCON ENTERTAINMENT. Blade Runner 2049 [online]. In: . [cit. 2022-04-20]. Dostupné z: <https://s01.sgp1.digitaloceanspaces.com/inline/852503-vstmgojpyh-1506794141.jpg>
- [13] KRAUZE, Eva a Daniel SHAKE. Мы - Возможно (Official Video). YouTube [online]. 8. 9. 2017 [cit. 2022-04-20]. Dostupné z: <https://www.youtube.com/watch?v=B1yIJ706i78>
- [14] ABUOLAIM, Abdullah, Mauricio DELBRACIO, Damien KELLY, Michael S. BROWN a Peyman MILANFAR. Learning to Reduce Defocus Blur by Realistically Modeling Dual-Pixel Data [online]. 2021 [cit. 2022-04-19].
- [15] KOTERA, Jan. Image Deblurring in Demanding Conditions [online]. Prague, 2020 [cit. 2022-04-19]. Dostupné z: [http://staff.utia.cas.cz/kotera/phd\\_thesis.pdf](http://staff.utia.cas.cz/kotera/phd_thesis.pdf). Institute of Information Theory and Automation, Czech Academy of Sciences.
- [16] GONZALEZ, Rafael C. Digital Image Processing, Global Edition. Pearson Education Limited, 2017. ISBN 9780133356724.
- [17] Image restoration [online]. [cit. 2022-04-19]. Dostupné z: [https://cw.fel.cvut.cz/wiki/courses/b4m33dzo/labs/6\\_restoration](https://cw.fel.cvut.cz/wiki/courses/b4m33dzo/labs/6_restoration)
- [18] ZHANG, Shanghang, Xiaohui SHEN, Zhe LIN, Radomír MĚCH, Joao P. COSTEIRA a Jose M. F. MOURA. Learning to Understand Image Blur [online]. ISR - IST, Universidade de Lisboa, 2018 [cit. 2022-04-19]. Dostupné z: [https://openaccess.thecvf.com/content\\_cvpr\\_2018/papers/Zhang\\_Learning\\_to\\_Understand\\_CVPR\\_2018\\_paper.pdf](https://openaccess.thecvf.com/content_cvpr_2018/papers/Zhang_Learning_to_Understand_CVPR_2018_paper.pdf). Carnegie Mellon University.
- [19] SWAIN, Anisha. Noise filtering in Digital Image Processing. MEDIUM [online]. 2018 [cit. 2022-04-20]. Dostupné z: <https://medium.com/image-vision/noise-filtering-in-digital-image-processing-d12b5266847c>

- [20] ZAMIR, Syed Waqas, Aditya ARORA, Salman KHAN, Fahad Shahbaz KHAN, Ming-Hsuan YANG a Ling SHAO. Multi-Stage Progressive Image Restoration [online]. 2021 [cit. 2022-04-20]. Dostupné z: <https://arxiv.org/abs/2102.02808>. Mohamed bin Zayed University of AI.
- [21] HWANG, Jeff a You ZHOU. Image Colorization with Deep Convolutional Neural Networks [online]. 2016 [cit. 2022-04-20]. Dostupné z: [http://cs231n.stanford.edu/reports/2016/pdfs/219\\_Report.pdf](http://cs231n.stanford.edu/reports/2016/pdfs/219_Report.pdf)
- [22] SUVOROV, Roman, Elizaveta LOGACHEVA, Anton MASHIKHIN, et al. Resolution-robust Large Mask Inpainting with Fourier Convolutions [online]. 2021 [cit. 2022-04-20]. School of Computer and Communication Sciences, EPFL.
- [23] SARA, Umme, Morium AKTER a Mohammad Shorif UDDIN. Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study [online]. Bangladesh, 2019 [cit. 2022-04-20]. Dostupné z: [https://www.researchgate.net/publication/331435447\\_Image\\_Quality\\_Assessment\\_though\\_FSIM\\_SSIM\\_MSE\\_and\\_PSNR-A\\_Comparative\\_Study/fulltext/5c7926ca92851c695049cd1b/Image-Quality-Assessment-through-FSIM-SSIM-MSE-and-PSNR-A-Comparative-Study.pdf](https://www.researchgate.net/publication/331435447_Image_Quality_Assessment_though_FSIM_SSIM_MSE_and_PSNR-A_Comparative_Study/fulltext/5c7926ca92851c695049cd1b/Image-Quality-Assessment-through-FSIM-SSIM-MSE-and-PSNR-A-Comparative-Study.pdf). National Institute of Textile Engineering and Research.
- [24] ALAKUIJALA, J., R. OBRYK, O. STOLIARCHUK, Z. SZABADKA a L. VANDEVENNE. Guetzli: Perceptually Guided JPEG Encoder [online]. 2017 [cit. 2022-04-20]. Dostupné z: [https://www.researchgate.net/publication/314942913\\_Guetzli\\_Perceptually\\_Guided\\_JPEG\\_Encoder](https://www.researchgate.net/publication/314942913_Guetzli_Perceptually_Guided_JPEG_Encoder). Google Research Europe.
- [25] SHRIMALI, Kushashwa Ravi. Image Quality Assessment : BRISQUE [online]. 20. 6. 2018 [cit. 2022-04-20]. Dostupné z: <https://learnopencv.com/image-quality-assessment-brisque/>
- [26] LIANG, Jingyun, Jiezhang CAO, Guolei SUN a Kai ZHANG. SwinIR: Image Restoration Using Swin Transformer [online]. 2021 [cit. 2022-04-20]. Dostupné z: <https://arxiv.org/pdf/2108.10257.pdf>. Computer Vision Lab, ETH Zurich, Switzerland.
- [27] WAN, Ziyu, Bo ZHANG, Dongdong CHEN, Pan ZHANG, Dong CHEN, Jing LIAO a Fang WEN. Bringing Old Photos Back to Life [online]. Hong Kong, 2020 [cit. 2022-04-20]. Dostupné z: <https://arxiv.org/pdf/2004.09484.pdf>. City University of Hong Kong.

- [28] ZAMIR, Syed Waqas, Aditya ARORA, Salman KHAN a Munawar HAYAT. Learning Enriched Features for Real Image Restoration and Enhancement [online]. 2020 [cit. 2022-04-20]. Dostupné z: <https://arxiv.org/pdf/2003.06792.pdf>. Inception Institute of Artificial Intelligence, UAE.
- [29] PATEL, Ripal B., Kishor BAMNIYA a A.N. PATEL. Image Quality: Based On SSIM: Image Assessment. LAP LAMBERT Academic Publishing, 2014. ISBN 3659583561.

## 5 SEZNAM OBRÁZKŮ A TABULEK

Obrázek 1: Ukázka zhoršující se kvality se zvyšujícím se BRISQUE skóre .....	10
Obrázek 2: Příklad deblurringu SPZ.....	11
Obrázek 3: Blur obrázku se vstupními hodnotami: délka 45px; theta 15° .....	12
Obrázek 4: Příklad zesíleného ringing effectu.....	12
Obrázek 5: Příklad zostření Wiener filtrem .....	13
Obrázek 6: SSIM měření kvality obrazu .....	14
Obrázek 7: Využití GPU a VRAM .....	14
Obrázek 8: SSIM hodnoty Gaussovské filtrace .....	15
Obrázek 9: SSIM hodnoty Mediánového filtru na Sůl a pepř šumu.....	16
Obrázek 10: SSIM odstranění šumu pomocí MPRNet.....	17
Obrázek 11: Obarvení obrazu pomocí DeepAI .....	18
Obrázek 12: Image Inpainting pomocí LAMA.....	19
Obrázek 13: Upscale pomocí Topaz Gigapixel .....	19
Obrázek 14: Ukázka hodnot PSNR při kompresi .....	21
Obrázek 15: SSIM algoritmus .....	23
Obrázek 16: Tepelná mapa komprese.....	24
Obrázek 17: Vizualizace koeficientů .....	26
Obrázek 18: Příklad denoisingu pomocí MPRNet .....	28
Obrázek 19: Typy masek v inpaintingu.....	29
Obrázek 20: Příklad inpaintingu obrazu .....	29
Obrázek 21: Upscale pomocí SwinIR.....	30
Obrázek 22: Ukázkový příklad využití nástroje .....	31
Obrázek 23: Zesvětlení pomocí MIRNet.....	32
Obrázek 24: Parametry modelu .....	33
Obrázek 25: Hlavička programu .....	34
Obrázek 26: Definice a použití argumentů programu .....	35

Obrázek 27: Zpracování obrazu.....	35
Obrázek 28: Definice funkcí PSNR a SSIM.....	35
Obrázek 29: Vygenerované BRISQUE vektory .....	36
Obrázek 30: Hierarchie programu .....	36
Obrázek 31: Výpis programu.....	37
Obrázek 32: Postup instalace .....	38
Obrázek 33: Struktura složek.....	38
Obrázek 34: PTH soubor .....	39
Obrázek 35: Deblurring složka .....	40
Obrázek 36: Runtime CUDA error .....	40
Obrázek 37: Kód Main.py .....	41
Obrázek 38: Rozhodovací větev v Main.py.....	41
Obrázek 39: Chyba při implementaci SwinIR .....	42
Obrázek 40: Výsledný obraz.....	42
Tabulka 1: Typy šumu .....	15
Tabulka 2: Popis vzorce MSE .....	20
Tabulka 3: Popis vzorce PSNR.....	20
Tabulka 4: Vzorec pro závislost světelnosti v obrazu .....	21
Tabulka 5: Vzorec pro závislost kontrastu v obrazu.....	22
Tabulka 6: Vzorec pro finální strukturu .....	22
Tabulka 7: Vzorec pro porovnávání světelnosti .....	22
Tabulka 8: Vzorec pro porovnávání struktury .....	23
Tabulka 9: BRISQUE vzorec .....	25

## **6 PŘÍLOHY**