UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

MASKOVANIE OBRAZOVÝCH CHÝB ZACHOVÁVAJÚCE TEXTÚRY

(RIGORÓZNA PRÁCA)

MGR. IVANA ILČÍKOVÁ

BRATISLAVA, 2015

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA ALGEBRY, GEOMETRIE A DIDAKTIKY MATEMATIKY

MASKOVANIE OBRAZOVÝCH CHÝB ZACHOVÁVAJÚCE TEXTÚRY

(RIGORÓZNA PRÁCA)

MGR. IVANA ILČÍKOVÁ

BRATISLAVA, 2015

Čestne prehlasujem, že túto rigoróznu prácu som vypracovala samostatne s použitím uvedených zdrojov.

V Bratislave dňa .....

Ivana Ilčíková

Poďakovanie

Týmto by som chcela poďakovať prof. Ing. Jaroslavovi Polecovi, CSc za odborný dohľad, cenné pripomienky a usmernenia pri návrhu metódy a písaní rigoróznej práce.

Abstrakt

Maskovanie chýb je technika regulácie chýb schopná zmierniť efekty chýb v multimédiách s využitím iba správne prijatých dát. Predstavujeme efektívny a vysoko škálovateľný maskovací algoritmus pre textúrované farebné obrázky. Stratená oblasť je obnovená extrapoláciou textúry z okolitých oblastí logicky asociovaných na úrovni superpixelov. S navrhovanou metódou zostávajú zachované aj hrany nachádzajúce sa v stratenej oblasti. Porovnávame výsledky našich experimentov s výsledkami "state of the art" metód a demonštrujeme, že náš navrhnutý algoritmus beží oveľa rýchlejšie.

Kľúčové slová: maskovanie chýb; aproximácia kontúr; superpixel; segmentácia; extrapolácia; kódovanie obrazu

Abstract

Error concealment is an error control technique capable of mitigating the error effects on multimedia using only the correctly received data. We introduce an efficient and highly scalable concealment algorithm for textured colour images. The lost area is restored by texture extrapolation from the surrounding regions logically associated on the superpixel level. With the proposed method also edges located in the lost area are retained. We validate results of the experiments against state of the art methods and demonstrate that our proposed algorithm performs much faster.

Keywords: error concealment; contour approximation; superpixel; segmentation; extrapolation; image coding

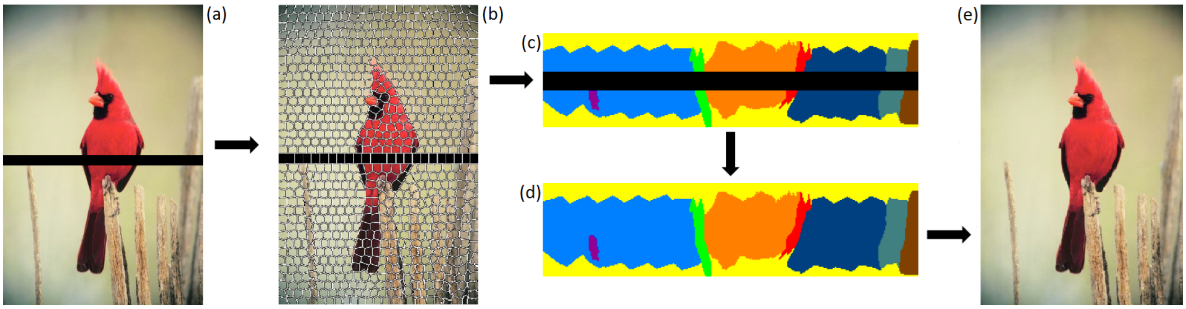
Obsah

Kapitola 1

Úvod

Prenos obrazu a videa vyžaduje rýchly a spoľahlivý prenosový kanál. Pri bezdrôtových sieťach je však chybovosť častým javom najmä v oblastiach so slabým pokrytím signálom alebo aj na miestach s príliš vysokým počtom zariadení. Pri prenose multimediálneho súboru prebieha kontrola správnosti prijatých dát prostredníctvom zvoleného poruchového režimu odolnosti (error resilience mode). Chyby v hlavičke musia byť opravené pomocou metódy automatického opakovania požiadavky (Automatic Repeat reQuest). V našej práci preto predpokladáme správne prijatie hlavičky a venujeme sa výlučne chybám v obrazových dátach. Najčastejšie používané režimy odolnosti pre obrazové dáta pracujú s blokmi [17, 46]. Chybné alebo chýbajúce obrazové dáta vždy produkujú poškodenie celých blokov alebo zhlukov blokov [45].

Prezentovaný algoritmus navrhuje novú techniku pre maskovanie chybných obrazových blokov s využitím základných princípov extrapolácie z nepoškodených susedných oblastí do poškodenej oblasti. Rýchle frekvenčno-selektívne metódy extrapolácie [31, 20] pracujú v rozsahu niekoľkých sekúnd. Používajú priamočiary prístup, ktorý extrapoluje každý pixel v rámci štvorcového okolia stratenej oblasti. Vysoko kvalitná rekonštrukcia obrazu je založená na komplexných patch-based [9] a group-based metódach [48], ktorých doba behu sa meria rádovo v minútach až hodinách. Náš navrhovaný algoritmus je zameraný na zlepšenie kvality pre rýchlu rekonštrukciu obrazu v časovom rozpätí niekoľkých sekúnd. Pre extrapoláciu vyberáme najrelevantnejšie pixely z okolia stratenej oblasti. Výber je realizovaný prostredníctvom segmentácie obrazu. Segmentácia poškodenej oblasti však chýba a je nutné ju doplniť. V našej práci predstavujeme nový algoritmus pre doplnenie segmentácie. Segmentácia určuje k sebe prislúchajúce časti obrazu v poškodenej oblasti a mimo nej. Textúru každej segmentovanej oblasti z nepoškodených častí obrazu aproximujeme zvlášť. Následnou extrapoláciou



Obr. 1.1: Navrhovaná metóda: (a) chybný vstupný obraz; (b) superpixely; (c) segmentácia oblastí susediacich so stratenou oblasťou; (d) zamaskované chyby segmentovaných oblastí; (e) opravený výstupný obraz.

získame doplnenie textúry daného segmentu v poškodenej oblasti. Extrapolácia v našom algoritme sa neobmedzuje iba na rýchlu Fourierovu tranformáciu, ale ako základ je možné zvoliť ľubovoľnú transformáciu, ktorá dokáže opakovať stochastické vlastnosti obrazu. Na obr. 1.1 poskytujeme prehľad krokov navrhovaného algoritmu.

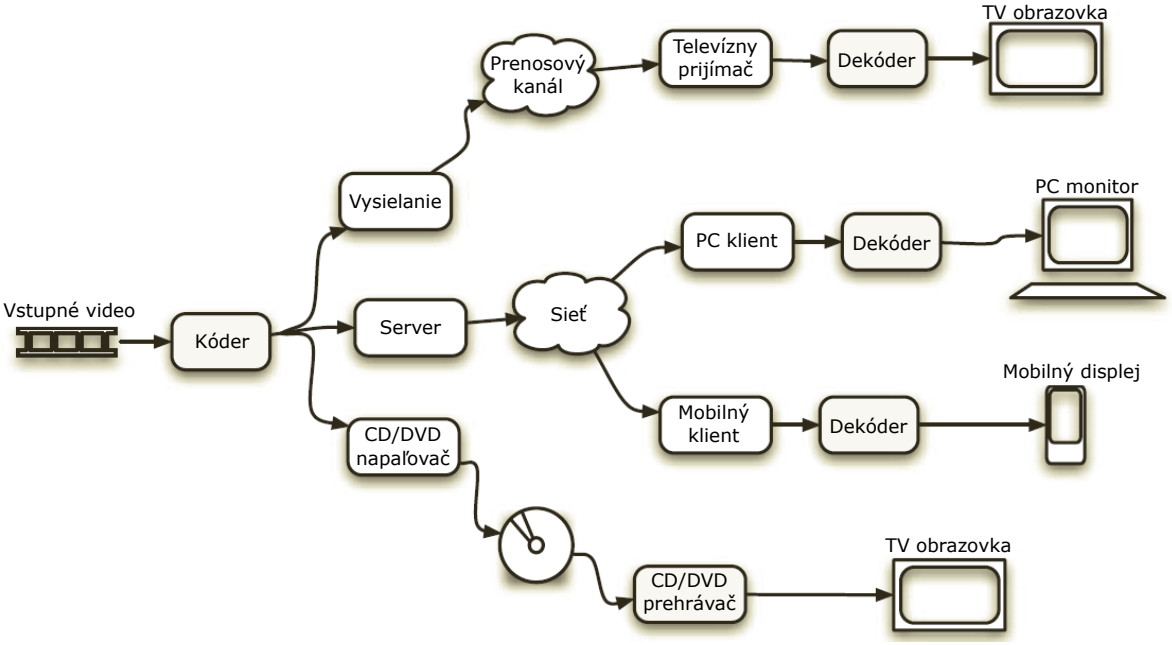
Zvyšok práce má nasledujúcu štruktúru. V kapitole 2 popisujeme proces a využitie kompresie obrazov a videa a následky výskytu prenosových chýb. Kapitola 3 sa venuje chybám odolnému kódovaniu obrazov a videa. V kapitole 4 analyzujeme existujúce metódy určené pre maskovanie obrazových chýb a pre doplnenie chýbajúcich častí obrazu (tzv. inpainting metódy). Na základe analýzy existujúcich metód sme navrhli vlastnú metódu, ktorej princíp popisujeme v kapitole 5. V ďalších troch kapitolách sa podrobne venujeme jednotlivým krokom navrhovaného algoritmu. Prvým krokom je segmentácia (kapitola 6). Najprv stručne popisujeme existujúce metódy pre nad-segmentáciu pomocou superpixelov a ďalej navrhujeme metódu s modifikovaným zlučovaním superpixelov, ktorá je schopná vysporiadať sa s poškodenými obrazmi. V kapitole 7 zavedieme koncept našej techniky maskovania poškodených tvarov (Shape Error Concealment) s mnohonásobným počtom tvarov v jednom obraze. Extrapolácia textúry segmentovaných a opravených oblastí je opísaná v kapitole 8. V kapitole 9, prezentujeme experimentálne výsledky a porovnávame navrhovaný algoritmus s existujúcimi metódami.

Kapitola 2

Kompresia obrazov a videa

Kompresia alebo kódovanie obrazov a videa je proces znižovania množstva dát potrebných na reprezentáciu digitálneho obrazového a video signálu pred prenosom alebo uložením. Komplementárna operácia - dekompresia alebo dekódovanie obnoví digitálny obrazový a video signál z jeho komprimovanej reprezentácie pred samotným zobrazením. Nespracované digitálne obrazové a video dáta (raw) majú tendenciu zaberať veľké množstvo úložnej pamäte alebo prenosovej kapacity. Preto kódovanie a dekódovanie obrazov a videa je nevyhnutné pre všetky aplikácie, v ktorých je kapacita pamäte alebo šírky prenosového pásma obmedzená. Takmer všetky spotrebiteľské aplikácie pre digitálny obraz a video spadajú do tejto kategórie, napríklad (obr. 2.1):

- *Digitálne televízne vysielanie:* televízne programy sú kódované pred prenosom cez pozemné, satelitné alebo káblové kanály s limitovanou šírkou pásma
- *Video na internete:* video je kódované a uložené na serveri, odtiaľ je prenášané (streamované) cez internet, dekódované na strane klienta a zobrazené
- *Streamovanie videa na mobilných telefónoch:* proces je podobný ako pre video na internete, ale kódované video sa prenáša cez mobilnú sieť, ako je EDGE, HSDPA alebo LTE
- *Video na digitálnych médiách:* video je kódované a uložené do digitálneho média (napr. Blu-ray, DVD, Flash disk), z ktorého potom číta digitálny prehrávač médií a dekóduje videá pre zobrazenie



Obr. 2.1: Kódovanie videa v rôznych typoch komunikácie [28]

Každý z týchto príkladov zahŕňa kodér, ktorý komprimuje a kóduje vstupný video signál do bitového toku a tiež dekodér, ktorý dekomprimuje alebo dekóduje bitový tok pre vytvorenie výstupného video signálu. Kódery alebo dekodéry sú často vstavané do zariadenia ako sú videokamery alebo video prehrávač.

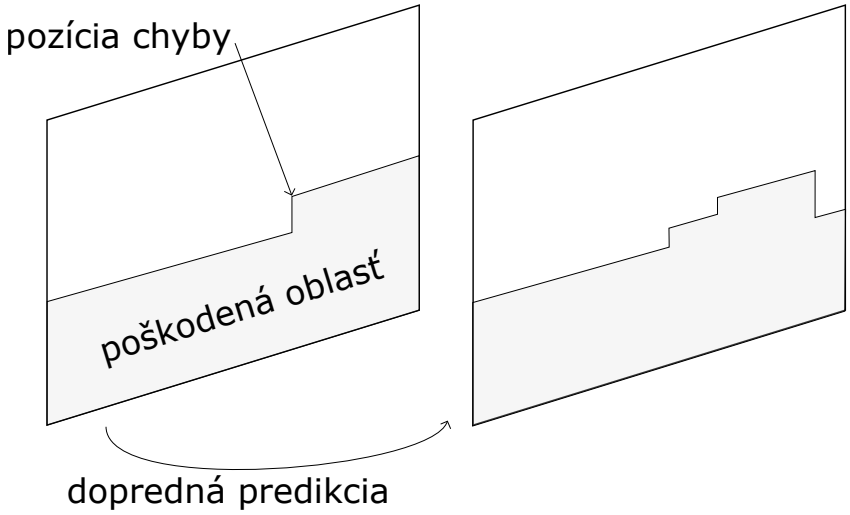
Kompresiu dát dosiahneme tým, že odstránime redundanciu, teda prvky, ktoré nie sú potrebné pre vernú reprodukciu dát. Niektoré typy dát obsahujú štatistickú redundanciu a môžu byť účinne komprimované pomocou bezstratovej kompresie tak, že rekonštruované obrazové dáta na výstupe dekodéra sú dokonalou kópiou pôvodných dát. Vo väčšine prípadov žiaľ bezstratová kompresia obrazu poskytuje príliš malé zníženie množstva dát. Vyššiu mieru kompresie možno dosiahnuť so stratovou kompresiou. Pri stratovej kompresii dekódované dáta nie sú totožné so zdrojovými dátami. Platí pravidlo, že čím vyššia kompresia, tým má dekódovaný obraz nižšiu vizuálnu kvalitu. Stratová kompresia je založená na princípe odstraňovania prvkov obrazu alebo videa, ktoré možno zahodiť bez toho, aby výrazne ovplyvnili vnímanie vizuálnej kvality divákom. Väčšina metód pre kódovanie videa odstraňuje priestorovú aj časovú redundanciu pre dosiahnutie kompresie. V časovej oblasti je zvyčajne vysoká korelácia (podobnosť) medzi snímkami videa, ktoré boli zachytené približne v rovnakom čase. Časovo susediace snímky (po sebe idúce zábery v časovom slede) sú často vysoko korelované, najmä ak je frekvencia snímok vysoká. V priestorovej oblasti je zvyčajne vysoká korelácia medzi pixelmi, ktoré sú blízko pri sebe, teda hodnoty susedných pixelov sú často veľmi podobné (obr. 2.2).

Obr. 2.2: Priestorová a časová korelácia vo video sekvencii [29]

2.1 Prenosové chyby

Komprimované video a obraz sú citlivé na prenosové chyby, ktoré sa často objavujú v kanáloch s nízkou prenosovou šírkou. Pre lepšie pochopenie, aké dôsledky majú neopravené prenosové chyby (ako sú napr. bitové chyby alebo strata paketov) na kvalitu dekódovaného obrazu, si môžeme zobrať kódovací systém videa. Kódovanie videa pozostáva z kompenzácie pohybu, prediktívnom kódovaní textúry a následnom entropickom kódovaní. Pri použití entropického kódovania chyby v bitovom toku spôsobia, že dekodér stratí synchronizáciu s kodérom. Kvôli strate synchronizácie sa objavia vizuálne artefakty v textúre zodpovedajúcej dekódovanej snímke. Rozlišuje dva typy šírenia chýb (obr. 2.3):

- **Priestorové šírenie chýb:** Keďže prediktívne kódovanie sa používa pre kódovanie textúry v snímke, chyby sa budú priestorovo šíriť do susedných pixelov. Bez akejkoľvek opravy chýb, môže dôjsť k poškodeniu veľkej časti snímky.
- **Časové šírenie chýb:** Keďže na kódovanie textúry po sebe idúcich snímok sa používa kompenzácia pohybu a časová predikcia textúry, chyby sa budú šíriť tiež na nasledujúce snímky.



Obr. 2.3: Časové a priestorové šírenie chýb

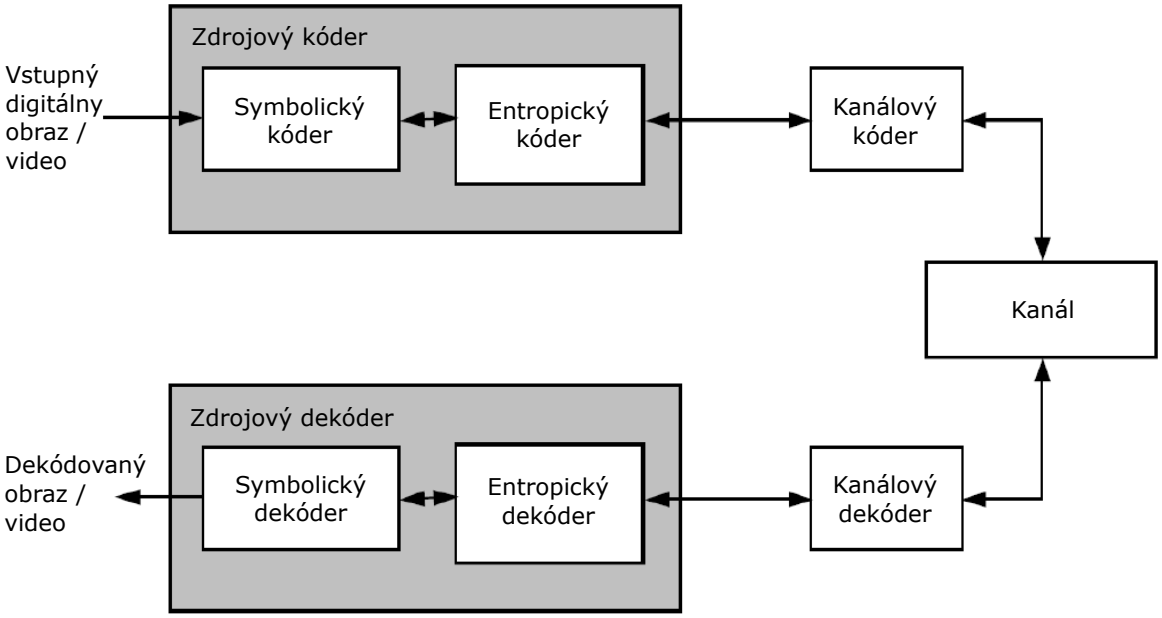
Kapitola 3

Chybám odolné kódovanie obrazov a videa

V komunikačnom systéme obrazov a videí (pozri zjednodušenú architektúru na obr. 3.1) je vstupný digitálny obraz, respektíve video najprv skomprimované zdrojovým kodérom, ktorý sa skladá zo symbolického kodéra a entropického kodéra. Symbolický kodér eliminuje väčšinu z bezvýznamnosti a redundancie, ktoré obrazové, respektíve video dáta v sebe majú. V entropickom kodéri sú následne využité štatistické vlastnosti kódovaných symbolov. Nakoniec je skomprimovaný dátový výstup zo zdrojového kodéra ešte kódovaný kanálovým kóderom, aby bol robustnejší voči na chyby náchylnému kanálovému prenosu. Na strane prijímača komunikačného reťazca sa vykonávajú inverzné operácie za účelom získania zrekonštruovaného obrazu, respektíve videa, ktoré má byť zobrazené. Na obr. 3.1 sú použité dvojité šípky, aby poukázali na skutočnosť, že môže existovať aj spätný kanál poskytujúci spätnú informáciu z dekodéra do kodéra. Ak by neexistoval žiadny spätný kanál, tak by boli použité iba jednoduché šípky smerujúce z kodéru k dekodéru.

Pre prenos komprimovaných obrazov a videa cez na chyby náchylné siete sú potrebné techniky odolnosti proti chybám. Existujú tri hlavné typy takýchto techník [34]:

1. **Chybám odolné zdrojové kódovanie** - Tieto techniky sa zaoberajú konverziou dostupných digitálnych obrazových a video dát do účinnej a odolnej digitálnej reprezentácie. Výberom chybám odolnejšej reprezentácie, zvyčajne za cenu menšej efektívnosti kompresie, môže byť minimalizovaný negatívny vplyv chýb. Jedným z problémov týchto techník je, že typicky ovplyvňujú kódovaciu syntax používanú kodérom a dekodérom



Obr. 3.1: Zjednodušená architektúra komunikačného systému obrazov a videí [34]

pre ich komunikáciu, a tá preto musí byť štandardizovaná. To zvyčajne znamená, že potom ako je definovaný štandard, sa už techniky odolnosti voči chybám nemôžu meniť alebo aktualizovať.

2. **Kanálové kódovanie a dekódovanie** - Tieto techniky zahŕňajú systematické vkladanie bitov navyše do bitového toku. Vložené bity neposkytujú nové zdrojové informácie, ale používajú sa na to, aby bolo možné odhaliť a opraviť chyby v bitovom toku. Kanálové kódovanie a dekódovanie je zvyčajne nezávislé na zdrojovom kódovaní a dekódovaní. Avšak spojenie zdrojového a kanálového kódovania môže priniesť významné zlepšenie. Dáta ako textúra, text alebo v prípade videa aj pohybové vektory môžu byť chránené rôznymi spôsobmi. Ich citlivosť na chyby je odlišná. Vzhľadom k tomu, že kanálové kódovanie zreteľne ovplyvňuje syntax, ktorú kodér a dekodér používajú pre komunikáciu, tak rovnako musí byť štandardizované. Avšak často je to vynechané zo štandardov kódovania obrazov a videa a je štandardizované na inom mieste (napr. v prenosových štandardoch). Táto možnosť poskytuje určitú pružnosť, pretože štandard kódovania obrazu alebo videa a prenosový štandard môžu byť vybrané samostatne.

3. **Maskovanie chýb** - Tieto techniky sa vyrovnávajú s chybami vnesenými prenosom cez kanál, ktoré neboli opravené procesom kanálového dekódovania alebo žiadnym iným prostriedkom, ako je napr. opakovaný prenos (ak bol k dispozícii). Techniky maskovania chýb spracovávajú a využívajú dostupné dekódované informácie (aj správne aj chybné), aby sa minimalizoval negatívny dopad chýb. Vzhľadom k tomu, že tento typ

techniky nemá žiadny vplyv na kódovaciu syntax alebo proces dekódovania, tak zvyčajne nepodlieha štandardizácii a je ponechaný ako jedna z otvorených oblastí pre súťaž a ďalšie zlepšenie.

Techniky chybám odolného dekóvania a maskovania chýb budú podrobnejšie popísané v nasledujúcich sekciách.

3.1 Chybám odolné dekóvanie a maskovanie chýb

Maskovanie chýb zahŕňa všetky techniky, ktoré umožňujú dekodéru minimalizovať negatívny dopad chýb pomocou dostupných poškodených a správne dekódovaných dát. Dekodér zvyčajne prechádza tromi krokmi spracovania:

1. **Detekcia chýb** - rozpozná, či došlo k nejakej chybe
2. **Lokalizácia chýb** - s najvyššou možnou presnosťou zistí, kde došlo k detegovanej chybe, respektíve chybám
3. **Maskovanie chýb** - znižuje negatívny dopad chýb, ktoré nemohli byť opravené kanálovým dekódovaním

3.1.1 Detekcia chýb

Prvým krokom v chybám odolnom dekódovaní je detekcia chýb. Jednou z techník detekcie chýb je detekcia syntaktickej nejednotnosti v zdrojovom kódovanom bitovom toku. Pre Huffmanovo kódovanie je najjednoduchším syntaktickým nesúladom výskyt nejakého nemožného, respektíve neplatného kódovaného slova. Je to také kódované slovo, ktoré nemôže byť začiatkom žiadneho platného kódovaného slova. Iný prípad syntaktickej nekonzistentnosti je neočakávaný výskyt markera v strede dát alebo jeho absencia, kde by sa očakávala, napr. na konci dátového bloku.

Kým detekcia chýb môže uľahčiť jeho maskovanie, nezdetegované chyby môžu byť skryté, napr. prostredníctvom postprocesingu (pozri Analýzu obrazu v sekcii 3.1.2.2) dekódovaného obrazu potom, ako boli použité techniky maskovania chýb. Väčšina informácií je kódovaná metódou kódovania s variabilnou dĺžkou (Variable Length Coding = VLC), zvyčajne buď

pomocou Huffmanovho alebo aritmetického kódovania. Keď dôjde k chybe, synchronizácia je takmer vždy stratená a chyby sa môžu šíriť po dlhú dobu. Z tohto dôvodu je dôležité zistiť chyby čo najskôr.

3.1.2 Lokalizácia chýb

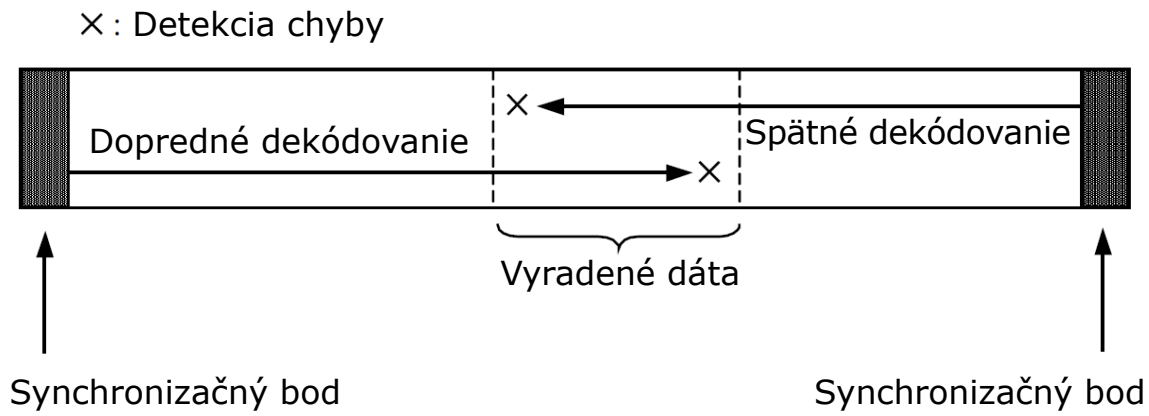
Presná lokalizácia chýb je základom pre ich zamaskovanie, pretože čím je lepšia lokalizácia chýb, tým menšie množstvo správnych dát sa musí zlikvidovať. Žiaľ väčšina techník pre lokalizáciu chýb je veľmi nepresných [34]. Detekcia syntaktickej nekonzistencie indikuje výskyt bitovej chyby alebo chyby kódovaného slova, ale zvyčajne príliš neskoro potom, ako ku chybe došlo. Za účelom dosiahnutia najlepších výsledkov lokalizácie chýb sa často používa niekoľko lokalizačných techník súčasne. V niektorých iných prípadoch je prijaté veľmi jednoduché riešenie predpokladajúc iba, že k chybe došlo N bitov alebo makroblokov pred tým, než bola detegovaná, kde N je stanovené na základe rozsiahleho experimentovania za príslušných podmienok. Niektoré techniky lokalizácie chýb sú podrobne popísané nižšie.

3.1.2.1 Chybám odolné entropické kódovanie

Lokalizácia chýb sa dá vykonať s využitím chybám odolného entropického kódovania. Napríklad obojstranné kódovanie s variabilnou dĺžkou (Reversible Variable Length code = RVLC) [36] môže byť použité pre ďalšiu lokalizáciu chýb, ako je to znázornené na obrázku 3.2. Ak sú použité RVLC kódy a dôjde k chybe, všetky dáta až do ďalšej značky sa preskočia a potom sa začne spätný dekódovací proces. Vďaka tomu je možné lepšie lokalizovať výskyt chyby a tým pádom aj minimalizovať vyradenie správnych dát.

3.1.2.2 Analýza obrazu

Vzhľadom k tomu, že chyby sú mnohokrát zaznamenané príliš neskoro po tom, čo sa objavili, niektoré poškodené textúrové dáta sú "správne" dekódované. To môže viesť k veľmi zvláštnym a viditeľným obrazovým artefaktom, ako sú napr. svetlo zelené 8×8 bloky zarovnané s 8×8 blokmi mriežky na tmavom pozadí. Tieto artefakty môžu byť detegované pomocou postprocesingových techník. Napríklad pre diskrétnu kosínusovú transformáciu (DCT) je na základe kódovacích schém možné zlepšiť lokalizáciu detegovanej chyby pri spracovaní susedných spojitostí. To znamená, že pri pohľade na silné obrazové nespojitosti zarovnané s



Obr. 3.2: Schopnosť lokalizácie chýb RVLC kódov [34]

Obr. 3.3: Snímka z video hovoru s ružovými 8 × 8 blokmi zarovnanými s 8 × 8 blokmi mriežky [34]

kódovacou mriežkou DCT, a za predpokladu, že je veľmi nepravdepodobné, že by sa silné a zvyčajne veľmi farebne odlišné hrany v obraze zhodovali s kódovacou mriežkou. Bloky, ktoré spĺňajú určité silné kritérium diskontinuity budú považované za poškodené.

Ďalší spôsob, ako odhaliť chyby textúry v postprocesing kroku, je definovať určité obmedzenia, napr. pokiaľ ide o sýtosť farieb. Napríklad v aplikácii pre video hovory, je rozumné predpokladať, že bloky 8 × 8 pixelov s vysokou sýtosťou farieb, napríklad ružovej alebo zelenej, sú nepravdepodobné (pozri obr. 3.3). Z toho dôvodu môžu byť také bloky tiež považované za chybné. Avšak kritériá, ktoré by mohli byť primerané pre jednu aplikáciu, nemusia byť pre inú. Napríklad, 8 × 8 bloky so sýtymi farbami by mohli existovať v aplikácii zahrňujúcej syntetický obrazový obsah. Techniky postprocesingu je v prípade potreby možné vypnúť, pretože vždy zahŕňajú určitú pravdepodobnosť odhalenia falošných chýb, aj keď len veľmi nízku.

3.1.3 Maskovanie chýb

Potom, čo sú chyby detegované a v čo najväčšej miere presne lokalizované, dekodér ich môže začať maskovať. Snaží sa minimalizovať negatívny dopad chýb v dekódovaných obrazoch len pomocou dostupných dekódovaných informácií. Proces maskovania je vždy založený na niektorých vedomých predpokladoch, ktoré môžu závisieť od typu aplikácie a od obsahu dát. V prípade videí v závislosti na dátach, ktoré sa používajú pre maskovanie chýb, techniky maskovania chýb môžu byť rozdelené do nasledujúcich troch kategórií [34]:

- **Priestorové maskovanie chýb** - na zamaskovanie chýb sa používajú dáta z aktuálneho časového okamihu
- **Časové maskovanie chýb** - na zamaskovanie chýb sa používajú dáta z iných (väčšinou predchádzajúcich) časových okamihov
- **Časovo-priestorové maskovanie chýb** - používajú sa údaje z aktuálneho časového okamihu a tiež ďalších časových okamihov

Tieto tri prístupy sú bližšie popísané nižšie.

3.1.3.1 Priestorové maskovanie chýb

Pri technikách priestorového maskovania chýb sa používajú iba dáta z aktuálneho časového okamihu. Poškodené oblasti v danom obraze sú získané interpoláciou dát z okolitých správne dekódovaných dát. Tento prístup môže mať vážne problémy v prípade, že poškodená oblasť obrazu predstavuje pomerne veľkú časť celého obrazu, a to najmä v prípade, že je obraz veľmi nehomogénny. To je dôvod, prečo sú tieto techniky väčšinou používané pre veľké obrazy, kde poškodené oblasti sú typicky relatívne malé v porovnaní s veľkosťou celého obrazu. Tieto techniky pracujú pomerne dobre pre homogénne oblasti a sú tiež vhodné pre sekvencie, kde je veľmi malá časová redundancia, ako je to v prípade prvej snímky video sekvencie po strihu scény alebo v prvej snímke úplne novej video sekvencie. Priestorové maskovanie chýb je tiež jedinou možnou metódou pre statické obrazy.

3.1.3.2 Časové maskovanie chýb

V časových maskovacích technikách dáta z iného časového okamihu, ako je aktuálny časový okamih, slúžia na vykonanie maskovania. Najbežnejší prístup je použitie dát z predchádzajú-

ceho časového okamihu, ale aj iné časové okamihy môžu byť použité. Ak sú tieto techniky použité pri sekvenciách s veľkými časovými redundanciami, čo zodpovedá väčšine prípadov, výsledky budú zvyčajne veľmi dobré. Ale ak sa dekodér pokúsi opraviť snímku v aktuálnom časovom okamihu pomocou údajov zo snímok v okolitých časových okamihoch, ktoré sú úplne odlišné, napríklad po strihu scény, tak sa môžu objaviť vážne problémy.

3.1.3.3 Časovo-priestorové maskovanie chýb

Myšlienka časovo-priestorových maskovacích techník je mať to najlepšie z priestorových a časových maskovacích techník. Niektoré časti obrazu môžu byť zamaskované pomocou priestorového maskovania, niektoré časti pomocou časového maskovania a ďalšie časti pomocou oboch metód. V ideálnom prípade, každé z týchto riešení by malo byť použité pre tie časti obrazu, ktoré sú priestorovo, časovo alebo priestorovo aj časovo veľmi homogénne, v danom poradí. K tomu musia byť použité zodpovedajúce techniky analýzy obrazu.

V ďalších častiach práce sa zameriame hlavne na priestorové techniky maskovania chýb, pretože tieto techniky sú ako jediné relevantné pre statické obrazy.

Kapitola 4

Existujúce metódy maskovania chýb obrazov

Rekonštrukcia obrazu je náročná úloha v mnohých multimediálnych aplikáciách, kde záleží na kvalite výsledného obrazu. Jedným z príkladov takej aplikácie je prenos obrazového alebo video signálu cez nespoľahlivý prenosový kanál, kde môže nastať strata paketov. Vypadnuté oblasti musia byť zamaskované použitím správne prijatých dát. Jeden z prvých algoritmov maskovania chýb navrhnutý v [32] je založený na metóde Markov random fields a zameriava sa na zachovanie vizuálne dôležitých čŕt a hrán, avšak nedokáže doplniť textúry.

Metóda [47] zavádza algoritmus blokovo založeného bilaterálneho filtrovania, ktorý rozširuje klasické bilaterálne filtrovanie operovaním na celých blokoch obrazu. Metóda má schopnosť zachytiť podobnosť na úrovni blokov a preto je vhodná pre maskovanie vypadnutých blokov. Táto metóda síce dokáže doplniť textúru skopírovaním celých blokov, no funguje dobre iba v prípade blokov, ktoré mali jednoliatu textúru. V prípade, že cez poškodený blok v originálnom obraze prechádzalo niekoľko hrán a obsahoval niekoľko rôznych textúr, tak po vyplnení vypadnutého bloku susedným blokom s inou štruktúrou môžeme jasne vidieť tvar bloku a teda výsledok maskovania nie je dobrý. Metóda má tiež problémy, ak za sebou vypadne niekoľko blokov, čo sa v praxi žiaľ stáva veľmi často.

Sekvenčné maskovanie chýb navrhnuté v [21] rekonštruuje stratenú oblasť pomocou riedkej lineárnej predikcie. Stratené oblasti sú vyplnené sekvenčne pomocou váhovanej kombninácie záplat, ktoré sú extrahované z dostupného 3D okolia. Techniku je preto možné efektívne použiť aj pre videá. Avšak aj táto metóda je vhodná iba pre maskovanie izolovaných blokov.

Alternatívny prístup pre maskovanie chýb obrazu je frekvenčno-selektívna extrapolácia (FSE) navrhnutá v [18]. Správne prijaté časti obrazu sú aproximované množinou bázických funkcií, ktoré sú definovavé na štvorcovej oblasti pokrývajúcej aj známe aj neznáme časti obrazu. Aproximácia správne prijatých dát je iteratívny proces minimalizovania chyby pomocou postupného výberu najdominantnejších bázických funkcií. FSE metóda je schopná konzistentne rozširovať signál (teda obraz) s rôznym obsahom ako napr. hladký obraz a obraz s hranami alebo s textúrou.

Neskoršia implementácia komplexnej FSE metódy [31] poskytuje podobnú kvalitu rekonštrukcie za desaťnásobne kratší čas. Táto technika vytvára model signálu z množiny Fourierových bázických funkcií, ktoré sú potom použité na nahradenie stratených častí obrazu.

Najnovšia úprava FSE metódy [20] vychádza z a priori informácie o nízko prechodovom správaní sa prirodzených obrazov. Bez tohto predpokladu sa v rekonštrukcii obrazu pomocou FSE metódy môžu vyskytovať vysoko frekvenčné artefakty. Pridaním nízko prechodového filtrovania zvyškovej chyby do iteratívneho procesu FSE metódy sa o niečo zlepšili výsledky, ale zároveň sa mierne zvýšila výpočtová náročnosť.

Problémom FSE metód je, že vždy aproximujú štvorcové okolie stratenej oblasti a teda často do aproximácie spadajú aj časti obrazu s nesúvisiacou textúrou. Aproximácia štvorcovej oblasti obsahujúcej niekoľko rôznych textúr je časovo náročnejšia ako aproximácia oblasti s jednou textúrou a výsledok nedosahuje dostatočnú kvalitu. Preto navrhujeme zahrnúť do procesu segmentačný krok, ktorý pomôže vyriešiť oba spomenuté problémy.

Ďalšou veľkou skupinou metód, ktoré síce primárne nie sú určené na maskovanie prenosových chýb, ale môžu byť na to použité, sú tzv. **inpainting metódy**. Inpainting metódy sa delia na dve hlavné skupiny:

1. *metódy založené na geometrických vlastnostiach obrazu* [23, 5, 37, 35] - umožňujú lokálne rozšíriť geometriu štruktúr v obraze na hraniciach oblasti, ktorú treba doplniť. Tieto metódy avšak nedokážu zreprodukovať textúry a preto nie sú vhodné na doplnenie väčších oblastí.

2. *textúrovo orientované metódy* [11, 7, 22, 9, 8, 48] - sú založené na kopírovaní a vkladaní textúry zo známych častí obrazu do oblasti, ktorú treba doplniť. Pri týchto metódach sa často stáva, že niektoré záplaty (patches) prinesú významovo nesúvisiaci alebo úplne nevhodný obsah do dopĺňanej oblasti.

Základná myšlienka jednej z prvých geometricky orientovaných inpainting metód [5] je hladko šíriť informácie z okolitých oblastí v smere izofót[1]. Najväčším problémom metódy je neschopnosť reprodukovať textúru väčších oblastí. Okrem toho, nevýhodou metódy je aj to, že výsledok získame až po niekoľkých minútach.

Ďalšia geometricky orientovaná metóda [37] vyhladzuje vstupný obraz jeho prevodom na vektorový obraz. Metóda má rôzne aplikácie, ako napr. zväčšenie obrazu, vyhladenie alebo odstránenie šumu a dá sa použiť aj na doplnenie menších poškodených oblastí.

Do skupiny geometricky orientovaných metód môžeme zaradiť aj metódu [35], ktorá prostredníctvom rozhrania aplikácie umožňuje používateľovi označiť dôležité štruktúry v obraze pomocou úsečiek alebo kriviek tak, aby bolo jasné, ako majú tieto štruktúry pokračovať v neznámej oblasti. Príkladom takej štruktúry môže byť okenný rám, rebrík alebo hranica horizontu, či hôr na obraze z prírody. Na jednu stranu je výhodou, že používateľ má možnosť svojím poznaním vylepšiť alebo aj upraviť výsledok doplnenia obrazu podľa jeho predstáv, no na druhú stranu sa už nejedná o čisto automatické doplnenie obrazu a preto nie je táto metóda vhodná pre opravu prenosových chýb.

Do druhej skupiny metód, ktoré sa v prvom rade zameriavajú na textúry, patrí napr. metóda syntézy textúr [11]. Táto metóda modeluje textúru ako Markov random field model. Predpokladom algoritmu je, že rozdelenie pravdepodobnosti hodnôt jasu pixela vzhľadom k hodnotám jasu jeho okolitej oblasti je nezávislé na zvyšku obrazu. Okolie pixela je modelované ako štvorcové okno okolo neho. Veľkosť okna je voľný parameter, ktorý určuje, nakoľko náhodne bude výsledná textúra vyzerať. Táto metóda je zameraná na čo najlepšie zachovanie štruktúry a dáva veľmi dobré výsledky pre celý rad syntetických aj reálnych textúr. Je možné ju použiť ako inpainting metódu v prípade, ak diera, ktorú treba doplniť, má byť celá vyplnená rovnakou textúrou. Pri zložitejších obrazoch s množstvom rôznych textúr však táto metóda neuspeje. Okrem toho je to veľmi pomalá metóda, keďže textúru syntetizuje po jednom pixeli.

Jedna z prvých inpainting metód [7] schopná plauzibilne doplniť väčšie a zložitejšie chýbajúce oblasti obrazu je založená na princípe patch-based modelovania obrazu. V porovnaní s predošlou metódou [11], ktorá postupuje po pixeloch, táto metóda vypĺňa chýbajúcu oblasť pomocou kopírovania väčších častí okolitého obrazu, tzv. záplat (patches). Tento prístup zvyšuje rýchlosť spracovania a tiež zlepšuje presnosť doplnených štruktúr. Okrem toho metóda zaviedla princíp sledovania významných hrán a ich primárne doplnenie. Tento princíp však funguje len pre rovné línie a nevie si poradiť so zaoblenými alebo zakrivenými črtami.

[1] Izofóta - čiara spájajúca miesta obrazu s rovnakým jasom.

Kľúčovou myšlienkou metódy [22] je, že záplaty (patches) dokážu zostať prirodzené aj po aplikovaní rôznych transformácií, ako napr. škálovania, otočenia alebo zmeny jasu a to sa snaží využiť pre získanie, čo najlepších výsledkov. Problémom je, že prehľadanie všetkých možností je výpočtovo veľmi náročná úloha a bez nastavenia základných obmedzení používateľom (ako napr. limitovanie škálovania alebo nepoužitie jasových transformácií) táto metóda môže bežať aj niekoľko hodín. Autori metódy preto navrhujú skúmať použitie rôznych optimalizačných metód k hľadaniu najlepších záplat a tiež použiť automatické metódy ako napr. detekciu opakujúcich sa elementov alebo symetrií.

Doposiaľ asi najlepšie vizuálne výsledky dosahuje Image Melding metóda [9]. Patrí do skupiny patch-based metód, pričom autori metódy pridali tri nové vylepšenia: Po prvé, obohatili priestor hľadania ďalšími geometrickými a fotometrickými transformáciami. Po druhé, integrovali obrazové gradienty do patch reprezentácie a nahradili zvyčajné farebné priemerovanie solverom Poissonovej rovnice. A po tretie, navrhli novú zmiešanú L_2/L_0 normu založenú na energii farieb a gradientov, ktoré vytvárajú stupňovité prechody medzi zdrojovými obrazmi bez toho, aby bolo treba obetovať ostrosť textúry. Tieto vylepšenia poskytujú okrem veľmi dobrých výsledkov pri zapĺňaní dier v obraze aj ďalšie možnosti využitia, ako napr. spájanie obrazov do panorámy, vkladanie objektov do iných obrazov s plynulými prechodmi, klonovanie objektov alebo harmonizáciu obrazu. Problémom metódy je, že je časovo veľmi náročná.

Daisy a kol. navrhli techniku priestorového blendovania [8] pre vylepšenie patch-based metód, ktoré iteratívne duplikujú bloky zo známych častí obrazu (záplaty) do oblastí, ktoré majú byť doplnené. Týmto procesom je možné doplniť veľké oblasti, pričom sa zachová aj textúra. Avšak v doplnenej oblasti sa často vyskytujú rôzne geometrické nezrovnalosti a artefakty. Metóda [8] je navrhnutá tak, že najprv vyhľadá takéto artefakty a následne ich zamaskuje. Výhodou je, že navrhnutá technika priestorového blendovania nezvyšuje výpočtovú náročnosť patch-based metódy. Autori článku [8] implementovali metódu [7] spolu s navrhnutou technikou blendovania ako voľný plugin do grafického softvéru GIMP, vďaka čomu je dostupná širokej verejnosti. Veľkou výhodou metódy je, že veľmi rýchla, aj keď nedosahuje také dobré výsledky ako Image Melding metóda [9].

Autori Group-based Sparse Representation (GSR) metódy [48] tvrdia, že tradičná patch-based riedka reprezentácia modelovania prirodzených obrazov zvyčajne trpí dvoma problémami: Po prvé, musí riešiť široko-spektrálny optimalizačný problém s vysokou výpočtovou zložitosťou v slovníkovom učení sa. Po druhé, každý patch je braný do úvahy samostatne v slovníku učenia a v riedkom kódovaní, čím sa ignorujú vzťahy medzi jednotlivými záplatami (patches), a to má za následok nepresné riedke kódovacie koeficienty. Preto navrhli metódu,

ktorá nahrádza záplatu (patch) ako základnú jednotku riedkej reprezentácie skupinou (group), ktorá sa skladá z nelokálnych záplat s podobnými štruktúrami. Okrem toho navrhli efektívnu metódu slovníkového učenia sa pre každú skupinu, čím dosiahli nižšiu zložitosť v porovnaní so slovníkovým učením sa z prirodzených obrazov. Táto metóda dosahuje skvelé výsledky ak oblasti, ktoré treba zaplniť, sú veľmi malé. Avšak v prípade väčších stratených oblastí je výsledok často príliš rozmazaný. Okrem toho je táto metóda napriek rôznym vylepšeniam stále veľmi pomalá. Na opravu jedného obrazu potrebuje niekoľko minút.

Kapitola 5

Maskovanie obrazových chýb zachovávajúce textúry

Na základe analýzy existujúcich metód pre maskovanie obrazových chýb sme identifikovali hlavné problémy a navrhli metódu, ktorá sa snaží riešiť tieto problémy. FSE metódy [18, 31, 20] aproximujú a extrapolujú štvorcové okolia vypadnutej oblasti a pritom nezohľadňujú často veľmi rozličný obsah štvorcovej oblasti. Takáto aproximácia rôznych textúr vedie k ich zmiešaniu a vo výslednej extrapolácii prináša neželané artefakty. Rovnako sa veľmi často stretávame s artefaktmi pri použití jednoduchých patch-based metód [7, 8], ktoré pre doplnenie chýbajúcej oblasti kopírujú z jej známeho okolia záplaty (patches). Toto zistenie nás viedlo k tomu, že sme navrhli do procesu maskovania chýb pridať segmentačný krok, ktorý pomôže zamedziť tvorbu artefaktov.

Jednotlivé kroky našej metódy sú znázornené na obr. 1.1. Na vstupe algoritmu je poškodený obraz - za sebou idúce chýbajúce bloky sú na obr. 1.1 (a) znázornené čiernou farbou. Prvým krokom algoritmu je nad-segmentácia poškodeného obrazu superpixelmi (obr. 1.1 (b)). Superpixely majú mať homogénnu farbu a textúru, podobnú veľkosť, majú byť pravidelne distribuované a ich okraje by mali sledovať významné hrany v obraze. Pre vytvorenie superpixelov existuje niekoľko efektívnych algoritmov, ktoré popisujeme v sekcii 6.1. Dôležitým faktom je, že vytvorenie superpixelov je možné urobiť v reálnom čase. Zlúčením podobných superpixelov získame segmentáciu obrazu. Na obr. 1.1 (c) sú zlúčené iba superpixely susediace s vypadnutou oblasťou a žltou farbou sú označené všetky ostatné superpixely. Nami upravený postup zlučovania superpixelov vhodný pre poškodené obrazy je popísaný v sekcii 6.2. Ďalším krokom algoritmu je doplnenie segmentácie v poškodenej oblasti (obr. 1.1 (d)). Pre túto úlohu sme navrhli originálny algoritmus, ktorý popisujeme v kapitole 7. Posledným

krokom našej metódy pre maskovanie obrazových chýb je doplnenie textúry, čím získame výsledný obraz (obr. 1.1 (e)). Postup doplnenia textúry popisujeme v kapitole 8.

Vďaka segmentácii máme obraz rozdelený na oblasti obsahujúce rovnakú alebo aspoň podobnú textúru. Textúru časti segmentu ležiacej v poškodenej oblasti doplníme extrapoláciou známej textúry toho istého segmentu, takže nedochádza k miešaniu textúr a k vzniku artefaktov. Okrem toho naša metóda dokáže veľmi dobre zachovať ostré hrany medzi susednými textúrami.

Jednotlivé kroky našej metódy bližšie popisuje v ďalších troch kapitolách.

Kapitola 6

Segmentácia pomocou superpixelov

Superpixely sú vytvorené zámernou nad-segmentáciou v obraze s cieľom vytvoriť malé kompaktné oblasti, ktoré môžu byť použité ako základné jednotky v nasledujúcich krokoch spracovania obrazu. Každý superpixel by mal ohraničovať homogénnu oblasť v zmysle farby a textúry. Navyše, okraj superpixelu by mal sledovať význačné hrany v obraze. Na druhú stranu, regulárna veľkosť a distribúcia superpixelov sú taktiež výhodné vlastnosti. Hlavným cieľom superpixelov je znížiť redundancie v obraze a tiež zefektívneniť výpočty.

Superpixely už boli použité v niekoľkých aplikačných oblastiach: segmentácia význačných objektov [16], sledovanie objektov (object tracking) [43], [49], detekcia význačných oblastí [24], video analýza a segmentácia [10], 3D rekonštrukcia z videa [40] atď.

6.1 Metódy vytvárania superpixelov

Doteraz už bolo navrhnutých množstvo spôsobov vytvárania superpixelov. Zoskupovací algoritmus (grouping algorithm) formulovaný ako problém delenia grafu (graph partitioning) bol navrhnutý v [27]. Tento algoritmus predstavil kritériá normalizovaného rezu (normalized cut) pre segmentáciu grafov.

Metóda konštantnej intenzity superpixelov *(Constant intensity superpixels method*) [41] optimalizuje superpixely na základe minimalizácie energie s využitím rezov grafu (graph cuts). Použitá funkcia energie uprednostňuje regulárne superpixely.

Felzenszwalbova efektívna segmentácia založená na grafoch [12] je populárna v komunite počítačového videnia. Algoritmus má jeden rozsahový parameter ovplyvňujúci veľkosť segmentu. Výsledná veľkosť a počet segmentov sa môže výrazne meniť v závislosti od lokálneho kontrastu.

Metóda známa ako *Quickshift* segmentácia obrazu [13] je založená na aproximácii pomocou kernelized mean-shift. Patrí k lokálnym mode-seeking algoritmom, využívajúcim LUV farbu a umiestnenie pixelu.

Jeden z najnovších algoritmov [4] je založený na morfologickom spracovaní obrazu. Využíva upravenú metódu predspracovania morfologickej šedotónovej rekonštrukcie predstavenej v [42]. Metóda je použitá na odstránenie nevýznačných hrán. Následne je aplikovaná segmentácia rozvodím (watershed) využívajúca sofistikovanú metódou generovania semienok (seeds).

Metóda nazvaná *Simple linear iterative clustering* (SLIC) [1, 2] je veľmi efektívna, ale popritom jednoduchá. Dosahuje veľmi dobré výsledky v krátkom čase behu. Algoritmus je široko používaný a často modifikovaný s cieľom dosiahnuť ešte lepší výkon. Rýchla verzia metódy SLIC implementovaná na GPU, sa nazýva gSLIC [26]. Nedávna metóda *Accelerated gSLIC* [6] umožňuje interaktívnu rýchlosť pre obrázky až do veľkosti 1280×960 pixelov. V tejto práci aplikujeme metódu SLIC ako dobrý kompromis medzi dobou behu a presnosťou.

6.1.1 SLIC metóda upravená pre použitie v poškodených obrazoch

SLIC algoritmus je založený na princípe *k*-means zhlukovania. Každý pixel je asociovaný s 5-dimenzionálnym vektorom $[L^*a^*b\ x\ y]$, kde L^*a^*b sú súradnice v CIE L^*a^*b farebnom priestore a x, y sú priestorové súradnice daného pixelu v obraze. L^*a^*b farebný priestor bol navrhnutý tak, aby farebné rozdiely namerané ako euklidovská vzdialenosť v L^*a^*b priestore korešpondovali s farebnými rozdielmi danými ľudským vnímaním. Aj keď používaný prevod z RGB do L^*a^*b vnáša konverzné chyby z dôvodu chýbajúcej informácie o spektrálnych vlastnostiach použitej kamery, táto chyba sa zdá byť irelevantná a použitím L^*a^*b súradníc je možné dosiahnuť lepšie výsledky v porovnaní s využitím RGB farebných súradníc.

Pri nepoškodených obrazoch je v inicializačnom kroku algoritmu rozmiestnených k počiatočných stredov zhlukov (klastrov) $C_i = [L_i^*\ a_i^*\ b_i^*\ x_i\ y_i]$ do pravidelnej mriežky s rozostupmi veľkosti $S = \sqrt{N/k}$, kde N je počet pixelov obrazu a k je požadovaný počet superpixelov.

V našom upravenom algoritme stredy zhlukov umiestňujeme vždy mimo stratenej oblasti, teda v závislosti od jej veľkosti inicializujeme menší počet superpixelov. Aj z tohto dôvodu by bolo ťažké odhadnúť, aký je optimálny počet superpixelov pre daný obrázok a teda na vstupe nášho algoritmu máme konštantu S, z ktorej sa odvádza počet superpixelov podľa upraveného vzorca $S = \sqrt{(N - N_p)/k}$, kde N_p je počet poškodených pixelov v obraze. Aj v nasledujúcich krokoch algoritmu vynechávame poškodené pixely.

Počiatočné stredy zhlukov posunieme v rámci ich 3×3 okolia do takého bodu, kde je najmenší gradient. Je to potrebné z dôvodu, aby sa zabránilo centrovaniu superpixela na hrane, a aby sa znížila pravdepodobnosť zasadenia superpixelu do zašumeného bodu.

Ďalším krokom je krok priradenia jednotlivých pixelov k niektorému zhluku. Na začiatku majú všetky pixely obrazu (okrem stredov zhlukov) nastavenú nekonečnú vzdialenosť k najbližšiemu stredu zhluku a k žiadnemu nie sú priradené. Postupne prechádzame cez všetky stredy zhlukov a pre každý pixel v oblasti veľkosti $2S \times 2S$ okolo stredu zhluku počítame jeho vzdialenosť D od práve prechádzaného stredu zhluku. Odvodenie výpočtu vzdialenosti D uvádzame v nasledujúcej sekcii 6.1.1.1. Ak je vypočítaná vzdialenosť menšia ako aktuálna hodnota vzdialenosti pixla, tak jeho vzdialenosť nastavíme na novú menšiu hodnotu a priradíme ho k danému zhluku.

Po prejdení cez všetky stredy zhlukov je už každý pixel obrazu priradený k takému zhluku, ktorého stred je k nemu najbližšie. Keďže do zhluku pribudli ďalšie body, tak je potrebné vypočítať nový stred. Vypočítame ho ako priemer $[L^*a^*b\ x\ y]$ vektora všetkých pixelov, ktoré patria do zhluku. Okrem toho potrebujeme aktualizovať aj zostatkovú chybu E, ktorú počítame ako L^2 normu medzi novým a starým stredom zhluku. Kroky priradenia a aktualizácie je možné opakovať iteratívne, až kým chyba začne konvergovať.

Na koniec je nutné vykonať ešte dodatočné spracovanie, aby sme zabezpečili kompaktnosť výsledných superpixelov. Detaily je možné nájsť v sekcii 6.1.1.2 Celý algoritmus okrem dodatočného spracovania pre zabezpečenie kompaktnosti superpixelov je zhrnutý v Algoritme 6.1.

6.1.1.1 Meranie vzdialenosti

Farba bodu je reprezentovaná v CIELAB farebnom priestore ako trojzložkový vektor $[L^*a^*b^*]$, ktorého rozsah možných hodnôt je známy. Poloha pixelu $[x\ y]$, na druhej strane, môže mať rozsah hodnôt, ktoré sa menia v závislosti od veľkosti obrazu.

Algoritmus 6.1 SLIC pre segmentáciu superpixelov v poškodených obrazoch

/* Inicializácia */
Inicializuj stredy zhlukov $C_k = \left[L_k^* \, a_k^* \, b_k^* \, x_k \, y_k\right]$ do pravidelnej mriežky
s rozostupmi veľkosti S mimo stratenej oblasti.
Posuň stredy zhlukov v rámci ich 3×3 okolia do bodu s najmenším gradientom.
Nastav označenie $l(i) = -1$ pre každý pixel i.
Nastav vzdialenosť $d(i) = \infty$ pre každý pixel i.
repeat

/* Priradenie */
for každý stred zhluku C_k **do**

for každý pixel i v $2S \times 2S$ oblasti okolo C_k **do**

Vypočítaj vzdialenosť D medzi C_k a i.
if $D < d(i)$ **then**

nastav $d(i) = D$
nastav $l(i) = k$

end if

end for

end for
/* Aktualizácia */
Vypočítaj nové stredy zhlukov.
Vypočítaj zvyškovú chybu E.

until $E \leq prah$

Definovať vzdialenosť D medzi pixelom i a stredom zhluku C_k v algoritme 6.1 ako jednoduchú euklidovskú vzdialenosť v 5-rozmernom $[L^*a^*b\ x\ y]$ priestore by spôsobilo nekonzistencie zhlukovania pre rôzne veľkosti superpixelov. Pri veľkých superpixeloch by priestorové vzdialenosti prevážili farebnú podobnosť, teda blízkosť v priestore by mala väčší relatívny význam ako podobná farba. Týmto spôsobom by sme vytvorili kompaktné superpixely, ktoré by ale nesledovali hranice v obraze. Pri menších superpixeloch by to bolo naopak.

Na to, aby sme mohli skombinovať dve vzdialenosti do jednej mierky, je potrebné ich normalizovať na základe ich maximálnych vzdialeností vo vnútri zhluku: N_c a N_s. Potom D' dostaneme nasledujúcim odvodením

$$d_c = \sqrt{(L_k^* - L_i^*)^2 + (a_k^* - a_i^*)^2 + (b_k^* - b_i^*)^2} \tag{6.1}$$

$$d_s = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{6.2}$$

$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2} \tag{6.3}$$

Autori metódy SLIC aproximujú maximálnu priestorovú vzdialenosť v rámci daného zhluku veľkosťou rozostupov v inicializačnej mriežke, teda definujú $N_s = S$. Stanovenie maximálnej farebnej vzdialenosť N_c nie je tak jednoduché, keďže farebné vzdialenosti sa môžu výrazne líšiť od jedného zhluku k druhému a od jedného obrazu k ďalšiemu. Autori sa rozhodli, že aj napriek tomu definujú hodnotu N_c pomocou konštanty m, ktorej na základe experimentov určili hodnotu 10. Takže rovnica výslednej vzdialenosti D je

$$D = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2} \tag{6.4}$$

V našich experimentoch sme prišli k záveru, že navrhnutá aproximácia maximálnej priestorovej a farebnej vzdialenosti v rámci jedného zhluku je príliš nepresná. Preto sme sa rozhodli vypočítať exaktne maximálne rozpätia všetkých piatich zložiek vektora $[L^*a^*b\ x\ y]$ a teda pracovať s upraveným vzorcom vzdialenosti D

$$D = \sqrt{\left(\frac{L_k^* - L_i^*}{N_L^k}\right)^2 + \left(\frac{a_k^* - a_i^*}{N_a^k}\right)^2 + \left(\frac{b_k^* - b_i^*}{N_b^k}\right)^2 + \left(\frac{x_k - x_i}{N_x^k}\right)^2 + \left(\frac{y_k - y_i}{N_y^k}\right)^2} \tag{6.5}$$

Algoritmus 6.2 Doplnená časť Priradenie z algortimu 6.1 o prepočet nových hodnôt N_L^k, N_a^k, N_b^k, N_x^k,N_y^k

```
/* Priradenie */
for každý stred zhluku C_k do
    for každý pixel i v 2S × 2S oblasti okolo C_k do
        Vypočítaj vzdialenosť D medzi C_k a i.
        if D < d(i) then
            nastav d(i) = D
            nastav l(i) = k
            vypočítaj nové hodnoty N_L^k, N_a^k, N_b^k, N_x^k,N_y^k
        end if
    end for
end for
```

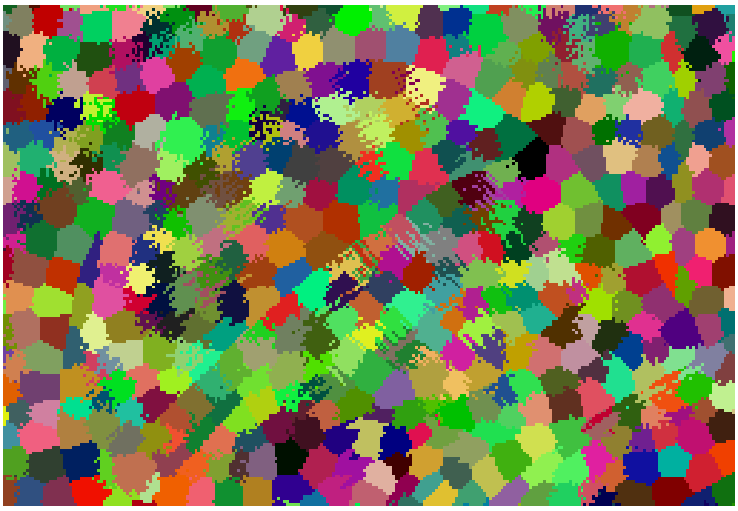
V algoritme 6.1 to predstavuje iba drobnú zmenu. Potom ako niektorý pixel priradíme do zhluku k, prepočítame nové hodnoty maximálnych rozpätí jednotlivých zložiek [*L*a*b x y*] vektora pre daný zhluk k. Doplnenú časť priradenia je možné vidieť v algoritme 6.2.

6.1.1.2 Zabezpečenie kompaktnosti superpixelov

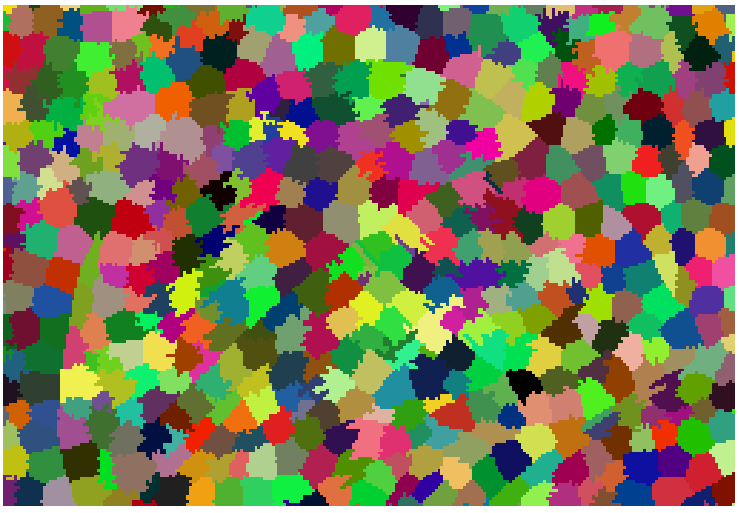
Výsledkom SLIC algoritmu je, že každý pixel v obraze bol priradený k niektorému semienku. Zatiaľ však nemáme informáciu o celistvosti jednotlivých množín (zhlukov) pixelov patriacich rovnakému semienku. Na to, aby sme získali takúto informáciu, postupne prechádzame všetky pixely Flood fill algoritmom. Začneme v ľavom hornom pixeli a Flood fillom vypĺňame všetky okolité pixely, ktoré sú priradené k rovnakému semienku ako prvý pixel. Po skončení Flood fillu, si do asociatívneho poľa, kde máme ako kľúče indexy semienok, vložíme pre príslušný kľúč vzniknutý superpixel. Tento superpixel nesie informáciu, ktoré všetky pixely doňho patria, koľko ich je a aká je priemerná *L*a*b* hodnota. Postupne prechádzame cez všetky pixely v obraze, ktoré ešte neboli Flood fill algoritmom priradené do žiadneho superpixelu. Po prejdení cez všetky pixely máme v asociatívnom poli pre každý kľúč uložený minimálne jeden superpixel. Kľúč, respektíve semienko, ktoré má priradené viac ako jeden superpixel, nemá kompaktný zhluk. Jeho zhluk je tvorený niekoľkými superpixelmi. Našim cieľom je, aby bol každý zhluk kompaktný (pozri obr. 6.1).

Pre každý nekompaktný zhluk si zoradíme všetky jeho superpixely od najväčšieho (s najväčším počtom pixelov) po najmenší. Najväčší superpixel necháme v danom zhluku a každý

(a) Vstupný obrázok - klobúk Lenny



(b) Nekompaktné zhluky



(c) Kompaktné zhluky = výsledné superpixely

Obr. 6.1: Porovnanie kompaktných a nekompaktných zhlukov

menší superpixel budeme chcieť priradiť do iného k nemu susednému zhluku. Všetky takéto menšie superpixely si zozbierame do jedného poľa S a utriedime ich od najmenšieho. Najmenšie superpixely budeme priraďovať do susedných zhlukov ako prvé. Našou snahou je priradiť superpixel k jeho farebne najpodobnejšiemu susedovi. Podobnosť dvoch superpixelov určujeme podľa vzťahu

$$D_{Lab} = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2} \tag{6.6}$$

kde $L_1^* a_1^* b_1^*$ je priemer $L^* a^* b^*$ hodnôt jedného superpixela a $L_2^* a_2^* b_2^*$ priemer druhého superpixela. Čím má D_{Lab} nižšiu hodnotu, tým sú dva superpixely farebne podobnejšie.

Keďže budeme chcieť priraďovať superpixely z nekompaktných zhlukov k ich farebne najpodobnejším susedom, tak v dátovej štruktúre, kde máme informácie o superpixeli (napr. ktoré pixely doňho patria, koľko ich je alebo, aká je priemerná *L*a*b* hodnota), si budeme uchovávať aj zoznam jeho susedov utriedených od najpodobnejšieho. Potom proces priradenia superpixela s k jeho najpodobnejšiemu susedovi t pozostáva z niekoľkých krokov. Do zoznamu pixelov superpixela t pridáme všetky pixely, ktoré patrili superpixelu s. Prepočítame priemernú *L*a*b* hodnotu superpixela t a aktualizujeme zoznam jeho susedov, teda pridáme mu do zoznamu všetkých susedov superpixela s, okrem neho samotného a vymažeme mu zo zoznamu susedov superpixel s, lebo ten už vlastne neexistuje. Nakoniec ešte požiadame susedné superpixely, aby si tiež aktualizovali zoznam svojich susedov. Tento proces priraďovania nekompaktných častí zhlukov k najpodobnejším susedom je prehľadne popísaný v algoritme 6.3. Výsledkom algoritmu je nízko úrovňová segmentáciu obrazu superpixelmi (pozri obr. 6.1c).

6.2 Zlučovanie superpixelov v poškodených obrazoch

K dosiahnutiu vysoko úrovňovej segmentácie obrazu musia byť podobné superpixely zlúčené (pozri obr. 6.2). Rozhodli sme sa použiť zlučovací algoritmus navrhnutý v [6], ale museli sme ho prispôsobiť, aby bol schopný zlučovať superpixely cez stratené oblasti. Algoritmus [6] je navrhnutý tak, že každý superpixel porovnáva so všetkými jeho susedmi a rozhoduje, či sa majú zlúčiť do jedného segmentu. Základnou myšlienkou nášho algoritmu je, že dva superpixely sú považované za susedné, aj keď sú oddelené stratenou oblasťou.

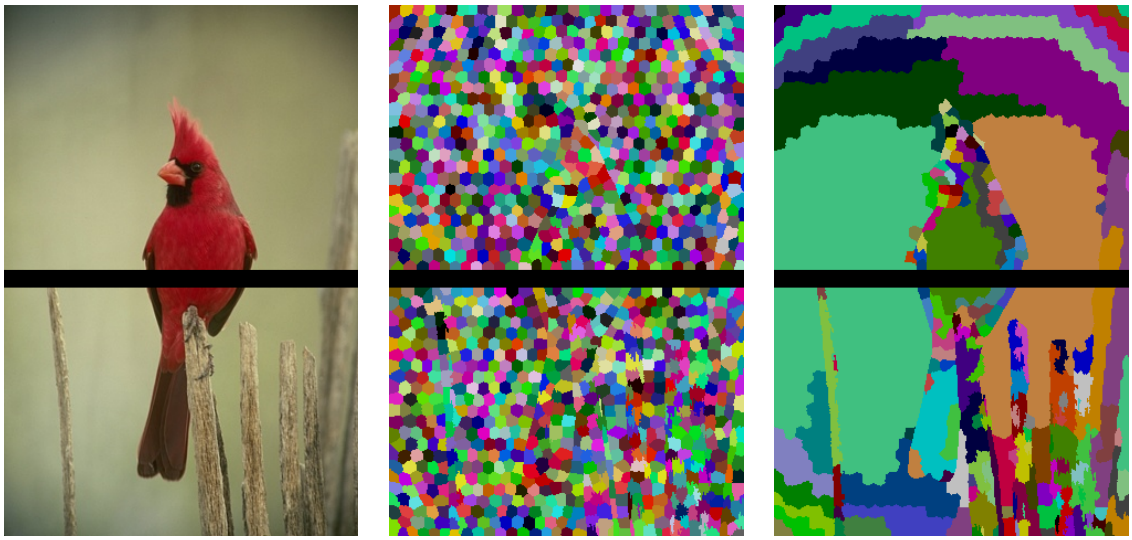
Pri implementácii pravidla, že superpixely sú susedné aj v prípade, že sú oddelené stratenou oblasťou, sme sa stretli s tým, že do jedného segmentu boli zaradené aj superpixely, ktoré

Algoritmus 6.3 Proces piraďovania nekompaktných častí zhlukov k najpodobnejším susedom

Utrieď pole superpixelov S, ktoré majú byť pričlenené do iného zhluku, od najmenšieho (s najmenším počtom pixelov).
for každý superpixel s z poľa S **do**

Pričleň superpixel s k nemu najpodobnejšiemu susednému superpixelu t.
Aktualizuj zoznam pixelov patriacich superpixelu t (pridaj do jeho zoznamu všetky pixely, ktoré patrili superpixelu s).
Prepočítaj priemernú L^*a^*b hodnotu superpixelu t.
Aktualizuj zoznam susedov superpixela t a utrieď ho od najpodobnejšieho suseda.
Požiadaj susedné superpixely, aby si tiež aktualizovali zoznam svojich susedov.

end for



(a) (b) (c)

Obr. 6.2: Proces segmentácie: (a) vstupný poškodený obraz; (b) nízko úrovňová segmentácia superpixelmi; (c) vysoko úrovňová segmentácia

boli od seba veľmi vzdialené (každý ležal na inom konci stratenej oblasti). Preto sme museli pridať podmienku, že vzdialenosť dvoch superpixelov, ktoré budeme považovať za susedné, nesmie prekročiť určitú dĺžku. Túto dĺžku sme určili ako 1,5-násobok veľkosti najdlhšej strany z kratších strán ohraničujúcich obdĺžnikov (boundig box) všetkých stratených oblastí v obraze.

Na začiatku procesu zlučovania superpixelov vytvoríme toľko segmentov, koľko je superpixelov. Do každého segmentu priradíme jeden superpixel ako prvý komponent súvislosti. Dátová štruktúra segmentu nesie rovnaké informácie ako dátová štruktúra superpixelu - sú tam informácie o tom, ktoré pixely patria do segmentu, koľko ich je, aká je priemerná *L*a*b* hodnota a zoznam susedných segmentov utriedených od najpodobnejšieho. V tomto prípade už medzi susedmi pribudnú aj susedia cez stratenú oblasť.

Postupne prechádzame cez všetky segmenty a zisťujeme, či ku práve prechádzanému segmentu s môžeme pričleniť niektorých jeho susedov. To, či k segmentu s pričleníme susedný segment t, závisí od ich vzájomnej farebnej podobnosti (danej vzťahom 6.6). Ak je hodnota D_{Lab} menšia ako používateľom zadaný prah, tak segment t pričleníme k segmentu s. To znamená, že celý zoznam komponentov segmentu t pridáme do zoznamu komponentov segmentu s. Ďalej pridáme aj všetky pixely segmentu t do zoznamu pixelov segmentu s a tiež prepočítame priemernú *L*a*b* hodnotu segmentu s. Okrem toho ešte aj aktualizujeme zoznam susedov segmentu s, teda do zoznamu susedov mu pridáme všetkých susedov segmentu t, okrem neho samotného a vymažeme mu zo zoznamu susedov segment t, lebo ten už vlastne neexistuje. Nakoniec ešte požiadame susedné segmenty, aby si tiež aktualizovali zoznam svojich susedov. Cyklus prechádzania cez všetky segmenty sa opakuje, kým sa v cykle nájdu aspoň dva segmenty, ktoré sa zlúčia, teda ktoré boli podobnejšie ako používateľom zadaná minimálna podobnosť. Po skončení cyklu zlučovania segmentov ešte musíme zlúčiť susedné komponenty patriace do toho istého segmentu. Je to potrebné z dôvodu, aby výsledné komponenty boli komponentmi súvislosti daného segmentu.

Po tom, ako skončíme proces zlučovania segmentov, už máme hotovú segmentáciu, ale je to iba segmentácia poškodeného obrazu. Ďalším krokom je doplnenie segmentácie v stratených oblastiach (pozri nasledujúcu Kapitolu 7). Na základe kompletnej segmentácie už budeme vedieť, ktoré textúry susediace so stratenou oblasťou máme extrapolovať do akej časti tejto stratenej oblasti. Proces extrapolácie textúr popisujeme v Kapitole 8.

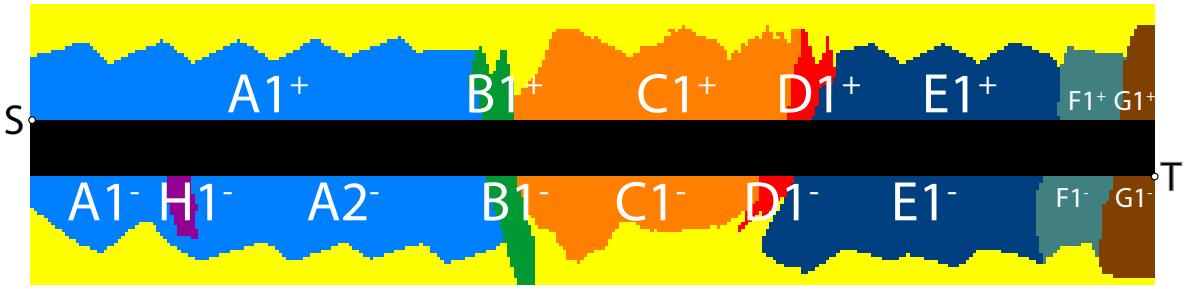
Kapitola 7

Maskovanie poškodených tvarov segmentov

Naším kľúčovým prínosom je rozšírenie techník pre maskovanie chýb hraníc jednotlivých objektov na segmentované obrazy, kde každý poškodený segment je považovaný za malý objekt, ktorý má byť zamaskovaný. Spojitosť susedných segmentov a častí segmentov rozdelených stratenou oblasťou na niekoľko kusov sú najväčšou výzvou.

Na obr. 1.1 (c), oblasti rovnakej farby patria tomu istému segmentu. Každý segment môže byť zložený z: jedného (napr. fialový segment) alebo viacerých (napr. svetlomodrý segment) komponentov súvislosti. Čierny pás uprostred označuje stratenú oblasť. Naším cieľom je zamaskovať ju spojením oblastí s rovnakou farebou, inými slovami, chceme spojiť komponenty každého segmentu (pozri príklad výstupného obrazu na Obr. 1.1 (d)).

Všeobecne možno povedať, že je pravdepodobnejšie, že segment bude mať skôr hladký obrys než aby mal ostré zmeny v jeho tvare. Preto je náš prístup k rekonštrukcii chýbajúcich kontúr založený na príbuznej metóde pre binárne obrazy známej ako maskovanie chýb tvarov (*shape error concealment*). Rôzne techniky[30, 33, 38] boli navrhnuté pre objektovo založené video, kde sú scény chápané ako kompozícia objektov [15]. Existujú tri typy metód pre maskovanie chýb tvarov v závislosti od typu informácií, ktoré sa používajú na vykonanie maskovania chýb: priestorové, časové a časovo-priestorové. V priestorových metódach maskovania chýb sa používajú iba dáta z aktuálneho časového okamihu. Z tohto dôvodu môžu byť použité nielen pre video sekvencie, ale tiež pre statické obrazy. Chýbajúce časti hraníc video objektov možno hladko doplniť pomocou aproximačných kriviek ako sú Hermitove splajny [30] alebo Bézierove krivky [33] alebo B-splajnové krivky [38].



Obr. 7.1: Písmenkové označenia a indexy poškodených segmentov získané sledovaním hranice stratenej oblasti. Segment A sa skladá z dvoch komponentov súvislosti, pričom spodný komponent má dve hraničné časti vzhľadom na stratenú oblasť.

Naša metóda maskovania chýb tvarov je rozšírením metódy navrhnutej v [38]. Upravili sme niektoré kroky, kde môžu byť použité dodatočné informácie z farebného vstupného obrazu, čím sme dosiahli urýchlenie algoritmu a tiež lepšie výsledky. V ďalších sekciách tejto kapitoly sú podrobne vysvetlené jednotlivé kroky algoritmu.

7.1 Indexovanie komponentov poškodených segmentov

V prvom kroku identifikujeme segmenty prerušené stratenou oblasťou a označíme ich pozdĺž hranice stratenej oblasti. Vyberieme dva body (S,T) stratenej oblasti – ľavý-horný a pravý-dolný. Potom sledujeme obrys poškodenej oblasti v smere hodinových ručičiek vychádzajúc z bodu S a smerujúc do bodu T a popri tom kontrolujeme pixely v 4-susednom okolí, či patria do ďalšieho komponentu súvislosti alebo časti toho istého segmentu. Keď narazíme na segment po prvýkrát označíme ho indexom 1^+. Pri narazení na rovnaký segment opätovný raz zvýšime index 2^+, atď. Takto každá časť hranice segmentu, ktorá susedí so stratenou oblasťou, má na obr. 7.1 značku pozostávajúcu z písmenkového označenia segmentu a indexového čísla. Pre pokrytie celej hranice stratenej oblasti sa aplikuje rovnaký postup aj proti smeru hodinových ručičiek s jednou malou zmenou, a to že priraďujeme záporné indexy.

7.2 Maskovanie jednoduchých segmentov

Segmenty, ktoré sú indexované len pozitívnymi alebo len negatívnymi indexmi, označujeme ako *jednoduché*. Na to, aby mohli byť zamaskované, nie je potrebné párovanie komponentov. Na obr. 7.1 fialový segment H je jediným jednoduchým segmentom. Pre doplnenie jeho chýbajúcej časti musíme v prvom rade odhadnúť jej tvar vnútri stratenej oblasti.

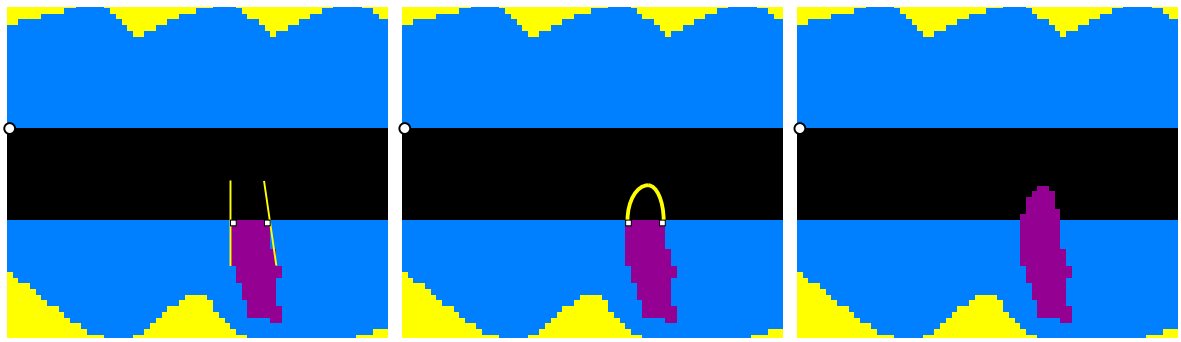
Najjednoduchší spôsob, ako spojiť koncové body kontúry, je použiť úsečku. Výsledkom by bola jednoduchá G^0 geometrická spojitosť kontúry. Vo väčšine prípadov je možné dosiahnuť lepšie výsledky cielením na vyšší stupeň spojitosti. Existujúce metódy [30, 33, 38] produkujú výsledky s C^1 spojitosťou vykonaním nasledujúcich krokov:

1. aproximácia známej kontúry v okolí koncových bodov dvojicou kriviek (jedna pre každý koncový bod)
2. nájdenie dotykových vektorov kriviek v oboch koncových bodoch
3. použitie dotykových vektorov pre konštrukciu krivky v poškodenej oblasti tak, že sa hladko napája na koncové body kriviek z kroku 1.

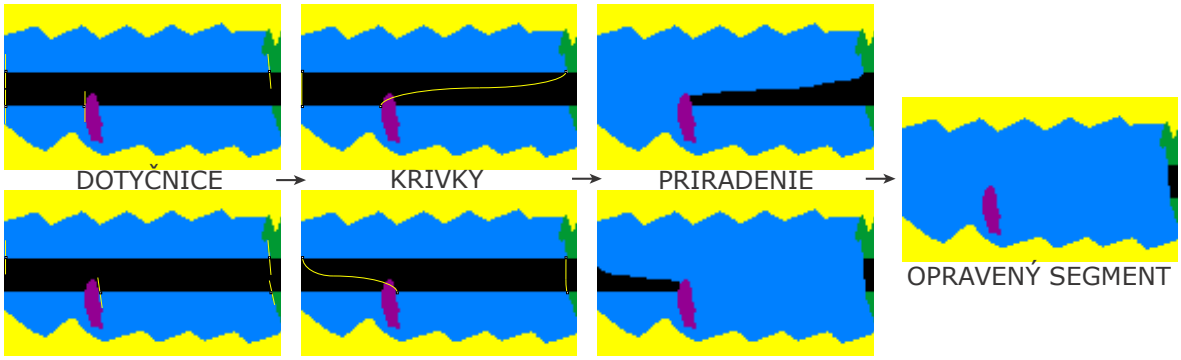
Rozhodli sme sa použiť rovnaký algoritmus pre všetky tri kroky, aký bol navrhnutý v [38]. B-splajnová krivka skonštruovaná pre doplnenie chýbajúcej kontúry je založená na T-splajnovej [3] reprezentácii nepoškodenej kontúry. T-splajnová krivka vytvára tvar zachovávajúcu aproximáciu a nemení vlastnosti pôvodnej kontúry. Takáto reprezentácia zaisťuje dobrý odhad dotykových vektorov v koncových bodoch. Čitateľ môže nájsť viac informácií v [38]. Po tom, ako bola skonštruovaná krivka v stratenej oblasti, priradíme ňou uzavretú oblasť k segmentu (pozri obr. 7.2). Rovnaký postup je aplikovaný na všetky jednoduché segmenty.

7.3 Maskovanie spárovaných segmentov

Akonáhle sú všetky jednoduché segmenty sú opravené, môžeme začať so spracovávaním segmentov skladajúcich sa z niekoľkých komponentov. Situácia je zložitejšia, pretože komponenty segmentu ležiace na rôznych stranách stratenej oblasti je potrebné spojiť. Pre každú



Obr. 7.2: Dotyčnice v koncových bodoch známej kontúry segmentu (vľavo) určujú tvar krivky v stratenej oblasti (v strede), ktorá je použitá pre doplnenie stratených pixelov segmentu (vpravo).



Obr. 7.3: Maskovanie chýb tvaru segmentu A z obr. 7.1 spája každý z jeho komponentov hraničiacich so stratenou oblasťou s najbližším komponentom z opačnej strany. V hornom riadku sú spárované komponenty $A1^+$a $A1^-$, v spodnom riadku sú spárované komponenty $A1^+$a $A2^-$. Konečný tvar segmentu je daný ako zjednotenie oboch čiastkových výsledkov.

poškodenú časť nájdeme najbližší komponent z druhej strany a spojíme ich koncové body použitím rovnakého algoritmu ako pre jednoduché segmenty. Koncové body sú prepojené tak, že spojovacie krivky sa nekrížia.

Všetky segmenty $B \ldots G$ na obr. 7.1 sa skladajú práve z dvoch hraničných komponentov, takže párovanie je priamočiare. Segment A sa skladá z častí označených $A1^+$, $A1^-$ a $A2^-$, čo vedie k nasledujúcim usporiadaným dvojiciam

$$\left(A1^+, A1^-\right), \left(A1^-, A1^+\right), \left(A2^-, A1^+\right).$$

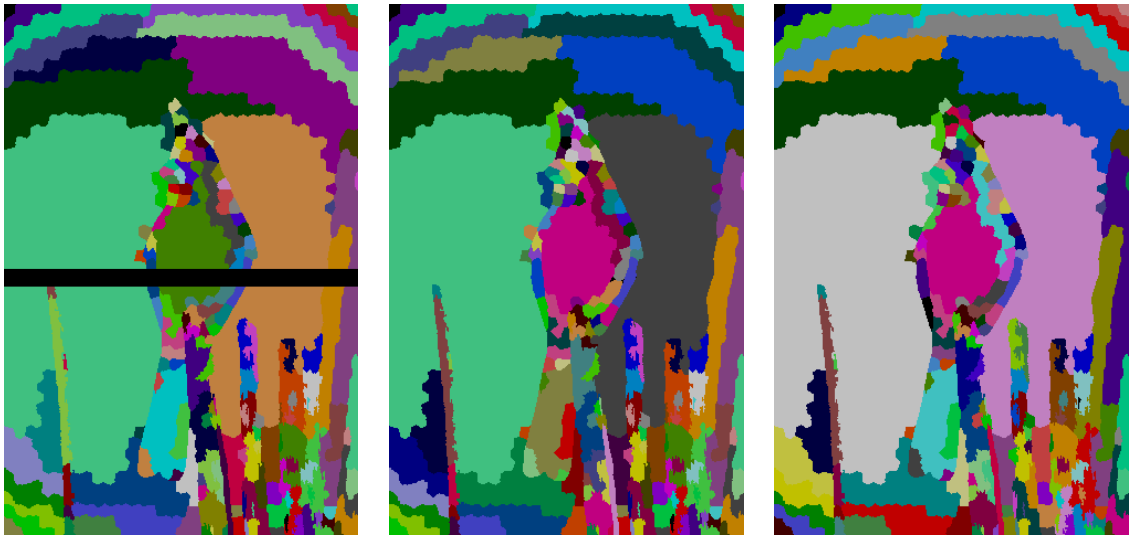
Po ich prevode do neusporiadaných dvojíc zostávajú

$$\left(A1^+, A1^-\right), \left(A1^+, A2^-\right).$$

Maskovanie chýb tvaru oboch dvojíc je znázornené na obr. 7.3. Prekrývanie jednotlivých oblastí nie je problém, pretože v konečnom kroku budú aj tak všetky pixely zaradené do toho istého segmentu.

7.4 Maskovanie zostávajúcich chýb

Predchádzajúce kroky neposkytujú záruku, že sa zamaskujú všetky chyby. V prípade, že sú na protiľahlých stranách stratenej oblasti poškodené jednoduché segmenty, niektoré pixely medzi nimi môžu zostať nepriradené k žiadnemu segmentu (pozri obr. 7.4). V poslednom kroku tohto postupu maskovania, sú také pixely iteratívne opravené použitím špeciálneho



(a) (b) (c)

Obr. 7.4: Doplnenie segmentácie v poškodenej oblasti: (a) vstupná segmentácia mimo poškodenej oblasti; (b) zamaskované jednoduché a spárované segmenty; (c) zamaskované zostávajúce chyby

modus filtra. Ak aspoň polovica susedov práve opravovaného pixela v jeho 9×9 okolí je už priradená k nejakému segmentu, potom tento pixel bude pridelený k najčastejšie sa vyskytujúcemu segmentu v jeho okolí.

Pôvodné komponenty segmentov vytvorených párovaním ležali na protiľahlých stranách stratenej oblasti. Preto tvar spojenia takýchto komponentov považujeme za optimálny. Na druhej strane, chýbajúci tvar jednoduchých segmentov je opravený s oveľa menšou istotou. Za účelom zlepšenia výsledkov maskovania modus filtra je pixel vždy priradený jednoduchému segmentu z jeho okolia, ak nejaký existuje. Po spracovaní všetkých segmentov získame odhadovanú segmentáciu stratenej oblasti.

Kapitola 8

Extrapolácia textúry

Vďaka oprave poškodených segmentov (popísanej v predošlej kapitole) už máme informáciu o tom, ktoré textúry kam extrapolovať a tým zamaskovať chyby poškodeného obrazu. Wang a kolektív [44] vyvinuli metódu pre aproximáciu neštvorcových oblastí. Textúra oblasti sa postupne aproximuje a potom sa oreže do tvaru segmentu. Nami navrhnutý algoritmus implementuje tento princíp pre doplnenie textúry poškodených segmentov. Chýbajúcu textúru segmentu odhadneme pomocou extrapolácie textúry z nepoškodených komponentov daného segmentu. Našou snahou je extrapolovať textúru tak, aby sme zachovali jej štruktúru a aby bol prechod do poškodenej oblasti, čo najmenej viditeľný.

8.1 Výber vhodnej bázickej funkcie

Nech A je segment predstavujúci oblasť ľubovoľného tvaru v diskrétnom 2D obraze s vnútornou štruktúrou textúry označenej ako $f(n_1,n_2)$. Najskôr pre segment A nájdeme opísaný obdĺžnik L. Potom, obdĺžnik L doplníme nulami tak, že jeho šírka a výška sú mocninami dvojky. To nám umožňuje použiť rýchle transformačné algoritmy. Veľkosť obdĺžnika je teraz $N_1 \times N_2$. Štruktúra textúry segmentu A vnútri obdĺžnika L je potom aproximovaná použitím vhodných bázických funkcií. Využívame množinu ortogonálnych funkcií $u_{k_1,k_2}(n_1,n_2)$, kde $k_1,n_1 \in \{0,1,\ldots,N_1-1\}$, $k_2,n_2 \in \{0,1,\ldots,N_2-1\}$. Aproximácia textúry pre obdĺžnik L po v krokoch je označená ako $g^{(v)}(n_1,n_2)$. Použitím lineárnej aproximácie ju môžeme vyjadriť ako súčet bázických funkcií ováhovaných vhodnými spektrálnymi koeficientmi

$$g^{(v)}(n_1,n_2) = \sum_{k_1,k_2 \in K_v} c^{(v)}_{k_1,k_2} u_{k_1,k_2}(n_1,n_2),$$

kde K_v označuje množinu indexov bázických funkcií použitých v $g^{(v)}(n_1,n_2)$ a $c^{(v)}_{k_1,k_2}$ označuje spektrálne koeficienty. Rozdiel medzi pôvodnou textúrou a jej aproximáciou je potom

$$r^{(v)}(n_1,n_2) = f(n_1,n_2) - g^{(v)}(n_1,n_2).$$

Teraz približne určíme tento rozdiel použitím vhodnej bázickej funkcie, aby sme mohli minimalizovať nasledujúcu chybovú funkciu

$$E_A = \sum_{(n_1,n_2)\in A} [f(n_1,n_2) - g(n_1,n_2)]^2,$$

kde A predstavuje pixely aproximovaného segmentu. Podľa [25] riešenie je dané maximalizovaním

$$\Delta E^{(v)}_A = \frac{\left[\sum_{(n_1,n_2)\in A} r^{(v)}(n_1,n_2)\, u_{k_1,k_2}(n_1,n_2)\right]^2}{\sum_{(n_1,n_2)\in A}\left[u^2_{k_1,k_2}(n_1,n_2)\right]}$$

Ďalšie podrobnosti sú uvedené v [25].

8.2 Extrapolácia chýbajúcich častí segmentu

Aproximáciu textúry správne prijatých častí segmentu iterujeme až kým nedosiahneme dostatočnú kvalitu. Kvalitu aproximácie môžeme merať napr. pomocou metriky PSNR (Peak Signal-to-Noise Ratio) alebo určením strednej kvadratickej chyby (Mean Squared Error). Po dosiahnutí stanovej kvality aproximácie je chýbajúca oblasť extrapolovaná ako príslušná časť obdĺžnika opísaného celému segmentu. Pri farebných obrazoch sa extrapolácia vykonáva samostatne pre všetky kanály farebného priestoru YCbCr.

Ak by pre opravu textúry nejakého strateného superpixelu boli použité všetky jeho bezprostredne susediace superpixely, niektoré z nich by mohli túto opravu narušiť. Naša metóda

(a) (b)

Obr. 8.1: Extrapolácia diskrétnou kosínusovou transformáciou (DCT) zo všetkých okolitých superpixelov (a) v porovnaní s extrapoláciou z oblasti vzniknutej zlúčením podobných superpixelov (b). Stredná kvadratická chyba (MSE) sa výrazne znížila po odstránení superpixelu nachádzajúceho sa najviac vľavo.

kontroluje kvalitu tým, že extrapolácia sa vykonáva len pre homogénne oblastí. Okrem toho, spracovanie homogénnych a menších oblastí vyžaduje menej času na dosiahnutie požadovanej hodnoty PSNR.

Na obr. 8.1 je červený superpixel opravený extrapoláciou superpixelov označených modrou farbou. Na ľavej strane (obr. 8.1a) superpixel najviac vľavo zodpovedajúci tmavému krídlu primiešal počas extrapolácie svoju textúru do rozmazaného pozadia. Na pravej strane (obr. 8.1b) odstránenie problematického superpixela vedie k takmer dokonalej rekonštrukcii. Presná segmentácia textúry je veľmi dôležitá, ale nie vždy možná pri našej metóde. Naša metóda zlyhá napríklad pri neostrých hranách, kde diskrétna segmentácia vnáša chyby na hranice segmentov, ktoré sa šíria ďalej pri oprave.

Kapitola 9

Experimentálne výsledky

Hlavným cieľom nášho výskumu je navrhnúť efektívnu metódu pre rýchle maskovanie obrazových chýb zachovávajúce textúry. Experimenty ukazujú, že prezentovaná metóda prekračuje kvalitu porovnateľne rýchlych frekvenčno-selektívnych extrapolačných metód [31, 20] a jednoduchých patch-based metód [8]. Naša metóda zároveň prekoná sofistikované patch a group-based metódy [9, 48] v čase potrebnom na dosiahnutie výsledkov porovnateľnej kvality. Pokusy boli vykonané na umelo degradovaných obrazoch vygenerovanými a lokalizovanými chybami na počítači s procesorom Intel Xeon X5550. Tabuľky 9.1 a 9.2 poskytujú porovnanie objektívnych chybových metrík a časovaní pre všetky prezentované výsledky. Použité štandardné chybové metriky poskytujú porovnanie s originálnymi, nepoškodenými obrazmi. Domnievame sa však, že maskovanie by malo byť tiež posudzované podľa subjektívneho dojmu z konzistencie obrazu bez ohľadu na pôvodný obraz.

9.1 Porovnanie s rýchlymi metódami

Frekvenčno-selektívne metódy extrapolujú každý pixel v rámci obdĺžnikového okolia stratenej oblasti. V dôsledku toho každý objekt v rámci zdrojového okna prispieva k frekvenčnej aproximácii. Nesúvisiace črty sa potom často miešajú medzi rôznymi regiónmi. Ak sú zdrojové pixely rekonštrukcie limitované na rovnaký segment a jeho tvar je zamaskovaný vopred, tón a textúra opravenej oblasti bude mať podobné vlastnosti. Porovnanie na obr. 9.1 a 9.2 ukazuje výkon rovnakej extrapolačnej metódy bez a s krokom segmentácie (obr. a, b, c). Pomocou segmentácie sme dosiahli lepšie subjektívne výsledky, ako je možné vidieť na oboch obrazoch a tiež lepšie objektívne výsledky, ako je možné vidieť v tabuľke 9.1. Vo

Tabuľka 9.1: Chybové a časové merania pre obraz vtáka kardinála (obr. 9.1), obraz Lenny (obr. 9.2) a obraz s raftom (obr. 9.3)

	Metóda maskovania	Metriky (celý obrázok)			Čas [min]
		MSE	PSNR	SSIM	
Vták kardinál	Bez segmentácie	**30.11**	**33.34**	**0.9320**	**1:41**
	Naša metóda	12.79	37.06	0.9735	0:04
	Metóda [8]	**10.89**	**37.76**	**0.9768**	**0:01**
Lenna	Bez segmentácie	79.57	29.12	**0.9265**	**2:43**
	Naša metóda	**36.87**	**32.46**	**0.9563**	0:11
	Metóda [8]	**103.20**	**27.99**	0.9415	**0:02**
Raft	Naša metóda	**37.13**	**32.43**	**0.9698**	**0:28**
	Metóda [8]	**40.57**	**32.05**	**0.9744**	**0:02**

všetkých troch kvalitatívnych metrikách: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR) aj Structural Similarity (SSIM) sme so segmentáciou dosiahli lepšie výsledky. Okrem toho aj čas potrebný na zamaskovanie chýb je kratší, pretože aproximácia textúry menších regiónov trvá kratší čas.

Implementácia rýchlej patch-based metódy [7] s pridaním techniky priestorového blendovania [8] do grafického softvéru GIMP poskytuje veľmi dobrý výsledok v prípade jednoduchých obrazov (pozri obraz vtáka kardinála (d) na obr. 9.1), ale pri oprave zložitejších obrazov (tvár Lenny obr. 9.2 (d) a obrázok s raftom obr. 9.3) môžeme vo výsledku vidieť niekoľko artefaktov.

9.2 Porovnanie so sofistikovanými inpainting metódami

Hlavná časť experimentov bola vykonaná za účelom porovnania výkonu a kvality maskovania našej metódy s najnovšími inpainting metódami: *Image Melding* [9] a *Group-based Sparse Representation for Image Restoration* (GSR) [48].

Pre obraz vtáka kardinála (obr. 9.4) Image Melding metóda jasne poskytuje najlepší výsledok. GSR metóda má problémy s pravým krídlom vtáka, ale kvalita meraná objektívnymi metrikami je aj tak o niečo lepšia v porovnaní s našou metódou. Na druhej strane, naša metóda potrebuje na zamaskovanie chýb iba štyri sekundy v porovnaní s niekoľko minútovými behmi ostatných dvoch metód. Okrem toho naša metóda poskytuje skvelé vizuálne výsledky. Iba zamaskovanie neostrej vetvy na pravej strane by sa mohlo zlepšiť.

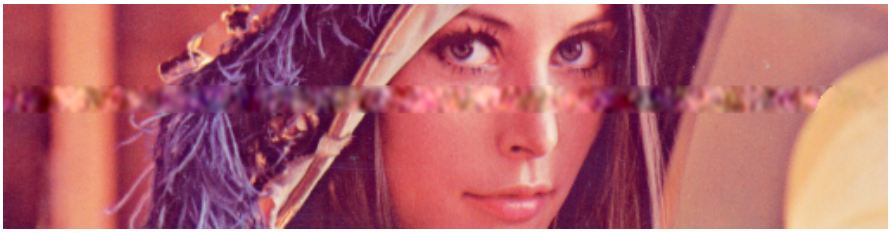
(a) Priama DCT extrapolácia z celého okolia vypadnutej oblasti

(b) Segmentácia

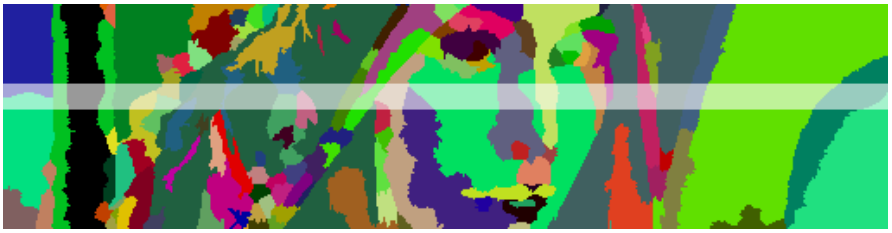
(c) Naša metóda s využitím segmentácie

(d) Metóda navrhnutá v [8] implementovaná do grafického softvéru GIMP [14]

Obr. 9.1: Porovnanie výsledkov DCT extrapolácie bez použitia a s použitím segmentácie pre obraz vtáka kardinála (a, b, c). Segmentácia minimalizovala artefakty extrapolácie textúry. Spodný obrázok poskytuje porovnanie s metódou implementovanou ako plugin do grafického softvéru GIMP. Tabuľka 9.1 poskytuje porovnanie objektívnych chybových metrík a časovaní pre všetky výsledky.



(a) Priama DCT extrapolácia z celého okolia vypadnutej oblasti



(b) Segmentácia

(c) Naša metóda s využitím segmentácie

(d) Metóda navrhnutá v [8] implementovaná do grafického softvéru GIMP [14]

Obr. 9.2: Porovnanie výsledkov DCT extrapolácie bez použitia a s použitím segmentácie pre obraz Lenny (a, b, c). Segmentácia minimalizovala artefakty extrapolácie textúry. Spodný obrázok poskytuje porovnanie s metódou implementovanou ako plugin do grafického softvéru GIMP. Keďže sa jedná o jednoduchú patch-based metódu, tak vo výsledku na tvári Lenny môžeme vidieť významné artefakty, ktoré vznikli kopírovaním záplat (patches) z nepoškodených častí obrazu. Tabuľka 9.1 poskytuje porovnanie objektívnych chybových metrík a časovaní pre všetky výsledky.

(a) Originál s vyznačeným poškodeným pásom

(b) Naša metóda

(c) Metóda navrhnutá v [8] implementovaná do grafického softvéru GIMP [14]

Obr. 9.3: Porovnanie výsledkov metódy [8] s našou metódou pre obraz s raftom. Priblížená časť obrazu ukazuje, že naša metóda poskytuje lepší vizuálny dojem, keďže do obrazu nepridala nesúvisiace časti z okolitých nepoškodených oblastí. Tabuľka 9.1 poskytuje objektívne chybové metriky a časové merania behu oboch metód.

(a) Originál s vyznačeným poškodeným pásom

(b) GSR metóda

(c) Image Melding metóda

(d) Naša metóda

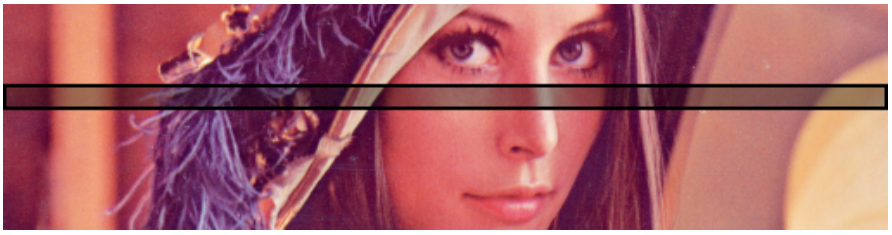
Obr. 9.4: Vizuálne porovnanie výsledkov pre obraz vtáka kardinála. Tabuľka 9.2 poskytuje objektívne chybové metriky a časové merania.

Tabuľka 9.2: Chybové a časové merania metód GSR, Image Melding a našej metódy

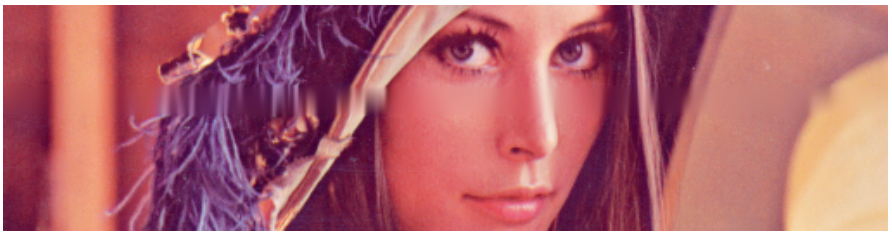
	Metóda maskovania	Metriky (celý obrázok)			Čas [min]
		MSE	PSNR	SSIM	
Raft	GSR	**26.54**	**33.89**	0.9754	**11:06**
	Image Melding	29.75	33.40	**0.9755**	10:14
	Naša metóda	**37.13**	**32.43**	**0.9698**	**0:28**
Lenna	GSR	28.95	33.51	0.9689	**11:18**
	Image Melding	**19.99**	**35.12**	**0.9749**	6:42
	Naša metóda	**36.87**	**32.46**	**0.9563**	**0:11**
Vták kardinál	GSR	6.52	39.99	0.9839	**6:30**
	Image Melding	**3.28**	**42.97**	**0.9873**	2:22
	Naša metóda	**12.79**	**37.06**	**0.9735**	**0:04**

Tvár Lenny (pozri obr. 9.5) má mnoho mäkkých prechodov, ale tiež jemné črty, ktoré sú ťažko segmentovateľné (pozri sekciu 8.2). Preto sa v prípade našej metódy na rekonštruovanej tvári objavili drobné artefakty. Inak sú to podobné výsledky ako pre obraz vtáka kardinála. Image Melding metóda má kvalitatívne najlepší vizuálny aj objektívkny výsledok (pozri Tabuľku 9.2). GSR metóda opäť rozmazala niektoré časti obrazu, napr. klobúk Lenny. Pri porovnaní časov behu jednotlivých metód je naša metóda ešte výraznejšie rýchlejšia, ako tomu bolo v prípade obrazu vtáka kardinála.

Obraz s raftom [19] obsahuje ako opakujúce sa textúry tak aj jedinečné objekty (pozri obr. 9.6). Naša metóda je 20× rýchlejšia a napriek tomu poskytuje porovnateľnú kvalitu obrazu v poškodenom páse. Dokonca textúru vody zrekonštruuje najlepšie, keďže v porovnaní s ostatnými metódami nie je rozmazaná.



(a) Originál s vyznačeným poškodeným pásom



(b) GSR metóda

(c) Image Melding metóda

(d) Naša metóda

Obr. 9.5: Vizuálne porovnanie výsledkov pre obraz Lenny. Tabuľka 9.2 poskytuje objektívne chybové metriky a čas behu jednotlivých metód.

(a) Originál s vyznačeným poškodeným pásom

(b) GSR metóda

(c) Image Melding metóda

(d) Naša metóda

Obr. 9.6: Výsledky porovnania pre obraz s raftom. Priblížená divoká vode ukazuje, že naša metóda poskytuje najlepší vizuálny dojem v najkratšom čase pre opravu tohto obrazu. Tabuľka 9.2 poskytuje objektívne chybové metriky a časové merania behu jednotlivých metód.

Kapitola 10

Záver a budúca práca

Doterajšie techniky maskovania tvarov predpokladali jediný objekt v stratenej oblasti a obmedzovali sa na jeho binárne rozlíšenie od pozadia. Hlavný prínos našej práce je rozšírenie maskovania tvarov na ľubovoľný počet objektov, ktorých vzájomné hranice vnútri poškodenej oblasti nie sú známe. Novú metódu sme prakticky aplikovali na opravu poškodených obrazov.

Navrhnutý algoritmus pre maskovanie chýb ukazuje nový prístup extrapolácie do stratenej oblasti v degradovaných obrazoch. Experimentálne výsledky jasne ukazujú kvalitatívne zlepšenie schopnosti maskovania chýb našej metódy prihliadajúcej na jednotlivé textúry v obraze v porovnaní s frekvenčno-selektívnymi extrapoláciami a jednoduchými patch-based metódami. Sofistikované patch-based a group-based metódy sú schopné poskytnúť lepšie výsledky, ale časová dĺžka ich behu nie je prípustná v reálnom čase. Článok s navrhovanou metódou [39] bol prijatý na medzinárodnú konferenciu IEEE Eurocon ’15.

Experimenty ukázali, že ťažko segmentovateľné časti obrazov s neostrými prechodmi vnášajú do výsledkov neželané artefakty. Preto sa chceme v našej ďalšej práci zamerať na možnosť použitia tzv. fuzzy segmentácie, teda segmentácie s neostrými hranami. Jeden pixel poškodenej oblasti by teda potenciálne mohol byť maskovaný z niekoľkých extrapolovaných segmentov.

Naším ďalším cieľom je zlepšiť výkonnosť algoritmu až na dosah interaktívnej obrazovej frekvencie tak, aby naša metóda mohla byť použitá aj pre videá a animácie. Jedno z možných urýchlení spočíva v rozdelení veľkých segmentov na menšie časti. Aproximácia textúry menších oblastí je rýchlejšia. Tu by sme tiež vedeli využiť fuzzy okraje pre rozdelené časti, aby sme umelým rozdelením segmentu nevniesli do rekonštrukcie neexistujúce hrany.

Po dosiahnutí interaktívnych časov behu algoritmu by v prípade videí bolo zaujímavé algoritmus rozšíriť aj na ďalšie snímky použitím supervoxelov namiesto superpixelov. Vďaka pridaným informáciám z predošlých snímok by sme dosiahli ešte lepšie výsledky maskovania prenosových chýb.

Literatúra

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels. *Technical report, École Polytechnique Fédrale de Lausanne, Report No. EPFL-REPORT-149300*, 2010.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[3] Gleb Beliakov. Shape preserving approximation using least squares splines. *Approx. Theory and its Appl.*, 16(4):80–98, 2000.

[4] Wanda Benešová and Michal Kottman. Fast superpixel segmentation using morphological processing. In *Proc. of Int. Conf. on Machine Vision and Machine Learning (MVML), 2014*. Avestia Publishing, 2014.

[5] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[6] Róbert Birkus. Accelerated gSLIC for Superpixel Generation used in Object Segmentation. In *Proc. of CESCG '15*, Apr. 2015.

[7] Antonio Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, Sept 2004.

[8] Maxime Daisy, David Tschumperlé, and Olivier Lézoray. A fast spatial patch blending algorithm for artefact reduction in pattern-based image inpainting. In *SIGGRAPH Asia 2013 Technical Briefs*, SA '13, pages 8:1–8:4, New York, NY, USA, 2013. ACM.

[9] S. Darabi, E. Shechtman, C. Barnes, D. B Goldman, and P. Sen. Image Melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. on Graphics*, 31(4):82:1–82:10, 2012.

[10] F. Drucker and J. MacCormick. Fast superpixels for video analysis. In *Workshop on Motion and Video Computing*, pages 1–8, December 2009.

[11] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038 vol.2, 1999.

[12] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *Int. J. C. Vision*, 59(2):167–181, September 2004.

[13] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *12th IEEE Int. Conf. on Comp. Vision*, pages 670–677, 2009.

[14] GIMP. http://www.gimp.org/. 20.9.2015.

[15] ISO. Information technology - coding of audio-visual objects - part 2: Visual. ISO/IEC 14496-2:2004. ISO, 2004.

[16] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Tie Liu, Nanning Zheng, and Shipeng Li. Automatic salient object segmentation based on context and shape prior. *BMVC*, 6:7, 2011.

[17] Li-Wei Kang and Jin-Jang Leou. A survey of error resilient coding schemes for image and video transmission based on data embedding. In *IEEE Asia-Pac. Conf. on Circ. and Sys., 2004*, volume 1, pages 145–148, Dec. 2004.

[18] A. Kaup, K. Meisinger, and T. Aach. Frequency selective signal extrapolation with applications to error concealment in image communication. *AEUE - International Journal of Electronics and Communications*, 59:147–156, 2005.

[19] Kodak. Lossless true color image suite. http://r0k.us/graphics/kodak/. accessed June 2015.

[20] J. Koloda, J. Seiler, A. Kaup, V. Sánchez, and A.M. Peinado. Frequency selective extrapolation with residual filtering for image error concealment. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 1976–1980, Florence, Italy, May 2014.

[21] J. Koloda, J. Østergaard, S. H. Jensen, V. Sánchez, and A. M. Peinado. Sequential error concealment for video/images by sparse linear prediction. *IEEE Transactions on Multimedia*, page 957–969, June 2013.

[22] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. Van Gool. Transforming image completion. In *British Machine Vision Conference (BMVC)*, August 2011.

[23] S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 259–263 vol.3, Oct 1998.

[24] P. Polatsek and W. Benesova. Bottom up saliency model generation using superpixels. In *Proc. of the 31st Spring Conf. on Comp. Graphics, Smolenice*, 2015.

[25] Jaroslav Polec, Tatiana Karlubíková, and Anton Březina. Hybrid orthogonal approximation of non-square areas. In *Comp. Vision and Graphics*, volume 32 of *Computat. Imaging and Vision*, pages 752–757. Springer, 2006.

[26] Carl Yuheng Ren and Ian Reid. gSLIC: a real-time implementation of SLIC superpixel segmentation. Technical report, Oxford University, 2011.

[27] Xiaofeng Ren and J. Malik. Learning a classification model for segmentation. In *9th IEEE Int. Conf. on Comp. Vision*, volume 1, pages 10–17, October 2003.

[28] Iain E. Richardson. *The H.264 Advanced Video Compression Standard*. John Wiley & Sons, Ltd, 2010.

[29] Iain E. G. Richardson. *H.264 and MPEG-4 Video Compression*. John Wiley & Sons, Ltd, 2003.

[30] G.M. Schuster, Xiaohuan Li, and A.K. Katsaggelos. Shape error concealment using hermite splines. *IEEE Transactions on Image Processing*, 13(6):808–820, June 2004.

[31] J. Seiler and A. Kaup. Complex-valued frequency selective extrapolation for fast image and video signal extrapolation. *IEEE Signal Processing Letters*, 17(11):949–952, November 2010.

[32] S. Shirani, F. Kossentini, and R. Ward. An adaptive markov random field based error concealment method for video communication in error prone environment. *Proceedings of ICIP*, 6:3117–3120, 1999.

[33] L.D. Soares and F. Pereira. Spatial shape error concealment for object-based image and video coding. *IEEE Transactions on Image Processing*, 13(4):586–599, April 2004.

[34] Luís Eduardo de Pinho Ducla Soares. *Error Resilience for Object-based Video Coding*. PhD thesis, Universidade Técnica De Lisboa, 2004.

[35] Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. Image completion with structure propagation. *ACM Trans. Graph.*, 24(3):861–868, July 2005.

[36] Y. Takishima, M. Wada, and H. Murakami. Reversible variable length codes. *Communications, IEEE Transactions on*, 43(234):158–162, Feb 1995.

[37] David Tschumperle and Rachid Deriche. Vector-valued image regularization with pdes: A common framework for different applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(4):506–517, 2005.

[38] E. Tsiligianni, L.P. Kondi, and A.K. Katsaggelos. Shape error concealment based on a shape-preserving boundary approximation. *IEEE Transactions on Image Processing*, 21(8):3573–3585, August 2012.

[39] Ivana Uhlíková, Wanda Benešová, Jaroslav Polec, and Tibor Csóka. Texture Aware Image Error Concealment. In *Proceedings of Eurocon '15*, pages 88–93. IEEE, September 2015.

[40] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Ben Ward, and Philip H. S. Torr. Videotrace: Rapid interactive scene modelling from video. In *ACM SIGGRAPH 2007 Papers*, New York, NY, 2007. ACM.

[41] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Comp. Vision - ECCV 2010*, volume 6315, pages 211–224. Springer, 2010.

[42] L. Vincent. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, April 1993.

[43] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *IEEE Int. Conf. on Comp. Vision*, pages 1323–1330, November 2011.

[44] Yao Wang, Qin-Fan Zhu, and L. Shaw. Maximally smooth image recovery in transform coding. *IEEE Transactions on Communications*, 41(10):1544–1551, October 1993.

[45] Vijitha Weerackody, Christine Podilchuk, and Anthony Estrella. Transmission of JPEG-coded Images over Wireless Channels. *Bell Labs Technical Journal*, 1(2):111–126, 1996.

[46] Stephan Wenger, G.D. Knorr, J. Ott, and F. Kossentini. Error resilience support in H.263+. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):867–877, November 1998.

[47] G. Zhai, J. Cai, W. Lin, X. Yang, and W. Zhang. Image error-concealment via block-based bilateral filtering. *IEEE International Conference on Multimedia and Expo*, page 621–624, June 2008.

[48] Jian Zhang, Debin Zhao, and Wen Gao. Group-based sparse representation for image restoration. *IEEE Transactions on Image Processing*, 23(8):3336–3351, Aug. 2014.

[49] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *IEEE Conf. on Comp. vision and pattern recognition*, pages 1838–1845, 2012.

Príloha

K rigoróznej práci prikladám DVD, ktoré obsahuje vytvorenú aplikáciu spolu s návodom na použitie. DVD obsahuje aj textovú časť rigoróznej práce vo formáte .pdf.