Proiect de semestru – Procesare Imaginilor

Nume, Prenume: Simina Dan-Marius

Grupa: 30232

Tema proiect: Completarea și restaurarea imaginilor cu metode patch-based

1. Specificatii detaliate

- Spcecificarea formatului datelor de intrare si de iesire
- Prezentarea temei, formulată exact, cu obiective clare și eventuale figuri explicative
- Detalierea cerintelor functionale ale aplicatiei

2. Analiză și fundamentare teoretică

Scopul acestui capitol este să explice principiile funcționale ale aplicației implementate. Aici descrieți soluția propusă din punct de vedere teoretic - explicați și demonstrați proprietățile și valoarea teoretică:

- algoritmi utilizați și/sau propuși,
- explicații/argumentări logice ale soluției alese,
- structura logică și funcțională a aplicației.

3. Proiectare de detaliu și implementare

Scopul acestui capitol este să documenteze aplicația dezvoltată în așa fel încât dezvoltarea și întreținerea ulterioară să fie posibilă. Cititorul trebuie să poată identifica funcțiile principale ale aplicației din ceea ce este scris aici.
Capitolul ar trebui sa conțină (nu se rezumă neapărat la):

- schema generală aplicației,
- descriere a fiecărei componente implementate, la nivel de modul, cu exemple de cod

4. Testare și validare

Demonstrarea functionarii corecte a aplicatiei prin prezentarea rezultatelor care sa demonstreze implementarea corecta a cerintelor /specificatiilor din punct de vedere calitativ/cantitativ.

1 Specificatii detaliate

1.1 Spcecificarea formatului datelor de intrare si de iesire

Date de intrare:

- **Imagine inițială**:
  - Format: .bmp
  - Dimensiuni: orice (dar recomandat sub 1080p pentru performanță)
  - Tip: imagine color (RGB)
- **Mască**:
  - Imagine binară (valori 0 și 255)

1.2 Regiunea ce trebuie completată (255) vs. regiunea păstrată (0)

Date de ieșire:

- **Imagine restaurată**:
  - Format: .bmp
  - Aceeași dimensiune și format ca imaginea inițială

1.3 Prezentarea temei, formulată exact, cu obiective clare și eventuale figuri explicative

Acest proiect are ca scop implementarea unei aplicații care să completeze automat o zonă lipsă într-o imagine utilizând tehnici de potrivire a fragmentelor (patch-uri). Algoritmul propus va analiza conținutul vizibil al imaginii și va identifica cele mai potrivite porțiuni pentru a reface regiunea deteriorată, menținând continuitatea vizuală și texturală.

Titlu:

“Completarea și restaurarea imaginilor cu metode patch-based”

Obiective:

Realizarea unei aplicații care:

1. Primește o imagine de intrare și o zonă lipsă (mască)
2. Completează regiunea lipsă folosind informații din restul imaginii (patch-based inpainting)
   - Algoritmul va analiza zonele din apropierea regiunii lipsă și va identifica patch-uri similare pentru a reconstrui conținutul.
   - Se vor compara regiuni ale imaginii utilizând metrici de similaritate.
   - Patch-urile selectate vor fi integrate în zona lipsă, asigurând o tranziție cât mai naturală.

1.4 Detalierea cerintelor functionale ale aplicatiei

1. **Încărcarea imaginii**:
    a. Utilizatorul selectează imaginea inițială.
2. **Ștergerea zonei de completat**:
    a. Zona de interes este marcată manual.
    b. Se generează o mască ce definește ce parte a imaginii lipsește.
3. **Completarea imaginii (Inpainting)**:
    a. Algoritmul caută patch-uri similare din imaginea neafectată
    b. Se completează zona mascată folosind cele mai potrivite patch-uri
4. **Salvarea rezultatului**:
    a. Imaginea restaurată este salvată.
    b. Se afișează comparația vizuală și metrică între imaginea originală și cea restaurată.

2 Analiză și fundamentare teoretică

2.1 Algoritmi utilizați în aplicația de completare a imaginilor

Aplicația implementează o metodă de completare a imaginilor bazată pe algoritmi patch-based, cu accent pe tehnicile PatchMatch. Principalii algoritmi utilizați sunt:

1. **Algoritm de căutare a celui mai apropiat vecin (Nearest Neighbor)** - Pentru fiecare patch din regiunea care trebuie completată, se caută cel mai similar patch din regiunea cunoscută a imaginii.
2. **Propagare** - Implementată în funcția propagate(), exploatează proprietatea de coerență a imaginilor, verificând dacă patch-urile vecine oferă potriviri mai bune.
3. **Căutare aleatoare ierarhică** - Implementată în funcția findBestMatch(), explorează succesiv spații de căutare restrânse, concentrându-se în jurul celor mai bune rezultate.
4. **Completare bazată pe prioritate** - Utilizează o coadă de priorități pentru a procesa mai întâi patch-urile importante.

2.2 Explicații logice ale soluției alese

Utilizarea PatchMatch:

- Oferă rezultate de calitate ridicată în timp rezonabil
- Exploatează coerența spațială a imaginilor naturale
- Combinarea propagării cu căutarea aleatoare evită minimele locale

Abordarea bazată pe prioritate:

- Asigură că structurile importante sunt completate primele:
- Patch-urile de la marginea regiunii de completat sunt procesate în ordinea priorității
- Prioritatea combină încrederea (cât de mulți pixeli cunoscuți conține) și informația de margine

Căutarea ierarhică:

- Reduce eficient spațiul de căutare.
- Începe cu explorare largă pentru a găsi regiuni promițătoare
- Rafinează căutarea în jurul celor mai bune potriviri

Post-procesare cu filtru gaussian:

- Reduce artefactele și netezește tranziția între regiunea originală și cea completată.

2.3 Structura logică și funcțională a aplicației

Interfața utilizator:

- Selectarea imaginii
- Selectarea regiunii de completat

- Vizualizarea rezultatelor
- Salvarea imaginii

Preprocesarea:

- Generarea măștii pentru regiunea selectată
- Identificarea patch-urilor de la frontieră

Procesul de completare implementat în imageReconstruction():

1. Inițializează harta de offseturi și masca
2. Calculează prioritatea pentru patch-urile de la frontieră
3. Adaugă patch-urile într-o coadă de priorități
4. Pentru fiecare patch (în ordinea priorității):
   i. Găsește cea mai bună potrivire prin propagare și căutare aleatoare
   ii. Completează patch-ul cu informația găsită
   iii. Actualizează masca și coada de priorități
5. Completează patch-urile rămase

Post-procesare:

Aplicarea filtrului gaussian

Această structură permite o completare eficientă a imaginilor, obținând rezultate vizual plauzibile prin prioritizarea structurilor importante și exploatarea coerenței imaginilor naturale.

3 Proiectare de detaliu și implementare

3.1 Schema generală a aplicației

Aplicația de completare a imaginilor urmează următorul flux de procesare:

1. **Încărcarea imaginii** și selecția regiunii de completat
2. **Pregătirea datelor** - generarea măștii pentru regiunea selectată
3. **Completarea imaginii** - căutarea celor mai bune patch-uri și reconstrucția regiunii
4. **Post-procesarea** - netezirea rezultatului final
5. **Salvarea rezultatului** - stocarea imaginii completate (funcționalitate viitoare)

[Încărcare Imagine] → [Selecție Regiune] → [Generare Mască] →

[Completare Imagine] → [Post-procesare] → [Salvare Imagine]

3.2 Descrierea componentelor implementate

3.2.1 Interfața utilizator

```
void MyCallBackFunc(int event, int x, int y, int flags, void* param);
```

Gestionează evenimentele mouse-ului pentru selectarea regiunii care trebuie completată. Funcția detectează când utilizatorul apasă, mișcă sau eliberează butonul mouse-ului, și actualizează coordonatele regiunii selectate corespunzător.

3.2.2 Generarea măștii

```
std::vector<std::vector<bool>> computeMask
(Mat img, int startX, int startY, int endX, int endY);
```

Creează o mască binară pentru regiunea selectată, unde true reprezintă pixelii ce trebuie completați. Funcția ajustează limitele regiunii pentru a evita depășirea marginilor imaginii și ține cont de dimensiunea patch-urilor.

```
bool isValidPatch(const std::vector<std::vector<bool>>& mask, int x, int y);
```

Verifică dacă un patch centrat în (x,y) nu conține pixeli din regiunea mascată, fiind astfel valid pentru a fi utilizat ca sursă.

3.2.3 Calculul priorității pentru patch-uri

```
double computePriority
(const std::vector<std::vector<bool>>& mask, int x, int y);
```

Determină prioritatea unui patch bazată pe două componente: încrederea (câți pixeli cunoscuți conține) și termenul de date (prezența marginilor). Patch-urile cu prioritate mai mare sunt procesate primele.

```
bool isBoundaryPatch(const std::vector<std::vector<bool>>& mask, int x, int y);
```

Verifică dacă un patch se află la frontiera dintre regiunea mascată și cea nemascată, fiind astfel candidat pentru completare.

3.2.4 Căutarea celor mai bune patch-uri

```
std::pair<int, int> propagate(const Mat& img, const
std::vector<std::vector<bool>>& mask, int x, int y,
std::vector<std::vector<std::pair<int, int>>>& offsetMap);
```

Exploatează coerența locală a imaginii verificând dacă offset-urile utilizate de vecinii patch-ului curent pot oferi potriviri mai bune.

```
std::vector<std::pair<int, int>> generateRandomPairs(const
std::vector<std::vector<bool>>& mask, int searchStartX, int searchEndX, int
searchStartY, int searchEndY, int step);
```

Generează perechi aleatorii de coordonate în spațiul de căutare pentru a explora eficient potențiale potriviri.

```
std::pair<int, int> findBestMatch(const Mat& img, const
std::vector<std::vector<bool>>& mask, int x, int y,
std::vector<std::vector<std::pair<int, int>>>& offsetMap);
```

Combină propagarea și căutarea aleatoare pentru a găsi cel mai bun patch pentru o poziție dată. Implementează o strategie de rafinare ierarhică a căutării.

```
int computeSSE(const Mat& img, const std::vector<std::vector<bool>>& mask, int
x1, int y1, int x2, int y2, int bestSoFar);
```

Calculează suma pătratelor diferențelor (SSD) între două patch-uri, cu optimizarea de oprire timpurie când valoarea depășește deja cea mai bună potrivire găsită.

3.2.5 Algoritmul principal de completare

```
Mat imageReconstruction(Mat& img, int startX, int startY, int endX, int endY);
```

Coordonează întregul proces de completare. Inițializează datele necesare, calculează prioritățile, procesează patch-urile în ordinea priorităților și gestionează actualizeaza cozii de priorități pe măsură ce regiunile sunt completate.

3.2.6 Completarea unui patch

```
void completePatch(Mat& img, std::vector<std::vector<bool>>& mask, int x, int y,
                   std::vector<std::vector<std::pair<int, int>>>& offsetMap);
```

Completează un patch prin găsirea celei mai bune potriviri și copierea informației din regiunea sursă în regiunea destinație. Actualizează masca pentru a marca pixelii ca fiind completați.

3.2.7 Post-procesare

```
Mat postprocessing(Mat img, int startX, int startY, int endX, int endY);
```

Aplică un filtru gaussian pentru netezirea rezultatului final, reducând eventualele artefacte vizuale și îmbunătățind tranziția între regiunea originală și cea completată.

```
Mat performConvolutionOperation(Mat img, std::vector<std::vector<int>> kernel,
int startX, int startY, int endX, int endY);
```

Implementează operația de convoluție pentru aplicarea filtrelor. În cazul implementării noastre, se folosește un filtru gaussian 3x3.

3.2.8 Salvarea imaginii (funcționalitate viitoare)

```
void saveImage(const Mat& img, const std::string& filename);
```

Această funcționalitate permite utilizatorului să salveze imaginea procesată.