Proiect Tehnologii Web Tema3 - JavaScript - Image Processing

Potop Horia Ioan 323AA

Descrierea aplicației

Aplicația JavaScript pe care am dezvoltat-o este o aplicație interactivă care permite prelucrarea și afișarea imaginilor preluate dintr-o sursă externă, folosind API-ul DogCEO (https://dog.ceo/api/breeds/image/random). Scopul principal al aplicației este de a aplica tehnici de procesare a imaginilor, cum ar fi un efect vizual de „mirror” și aplicarea operatorului Sobel pentru detectarea contururilor. Imaginile sunt preluate și procesate asincron pentru a nu afecta performanța interfeței utilizator. După procesare, imaginea este afişată pe un canvas HTML, iar timpii de execuție ai fiecărei operațiuni sunt măsurați și afişați într-un container de loguri pe pagina web, permițând utilizatorilor să urmărească performanţa aplicaţiei.

Partea teoretică

În domeniul procesării imaginilor digitale, tehnicile de detectare a contururilor și manipulare a pixelilor sunt esențiale pentru multe aplicații vizuale și de analiză. Un exemplu notabil este operatorul Sobel, care este utilizat pentru detectarea contururilor în imagini, aplicându-se pentru a calcula gradientul de intensitate al fiecărui pixel, în scopul de a identifica tranzițiile între zonele de intensitate diferită.

Operatorul Sobel funcționează prin aplicarea a două nuclee (matrice 3x3) care calculează derivata imaginii pe

direcțiile orizontală și verticală. Acesta este folosit de obicei pe imagini în tonuri de gri, pentru a marca tranzițiile de la un pixel la altul, accentuând contururile și marginile.

Tehnica „mirror” este o tehnică de manipulare a imaginilor, care inversează ordinea pixelilor pe axa verticală, creând un efect de reflexie. Aceasta este una dintre cele mai simple, dar populare tehnici de procesare vizuală, folosită frecvent pentru a adăuga un efect estetic sau de simetrie în imagini.

Descrierea implementării

1. **Preluarea imaginii din API:** Aplicația folosește funcția `fetchRandomDogImage()` pentru a obține un URL de imagine aleatorie din API-ul DogCEO. Acest URL este folosit pentru a încărca imaginea în aplicație prin funcția `loadImage()`, care gestionează încărcarea imaginii într-un obiect `Image`. Acest proces este asincron, ceea ce înseamnă că aplicația nu va bloca interfața utilizator în timp ce imaginea este încărcată.
2. **Afișarea imaginii pe canvas:** După încărcarea imaginii, aceasta este redată pe un canvas HTML cu dimensiuni fixe (500x500 px). Imaginea este scalată corect astfel încât să își păstreze proporțiile originale și este centrată pe canvas, pentru a oferi o experiență vizuală plăcută utilizatorului.
3. **Procesarea imaginii:** Aplicația oferă două opțiuni de procesare a imaginii:
    - **Mirror:** Această tehnică creează un efect de reflexie, procesând fiecare pixel de pe partea dreaptă a imaginii și inversându-i ordinea pe partea stângă.
    - **Sobel Operator:** Imaginea este mai întâi convertită într-o imagine în tonuri de gri, iar apoi se aplică

operatorul Sobel pentru a detecta contururile imaginii.

4. **Prelucrarea imaginii în secțiuni:** În loc de procesarea întregii imagini simultan, aceasta este împărțită în patru secțiuni (felii), fiecare fiind procesată asincron. Această metodă reduce timpul de procesare per segment, iar intervalele de 1 secundă între secțiuni permit utilizatorilor să observe procesul într-un mod vizualizat.
5. **Măsurarea timpului de execuție:** Fiecare etapă principală a aplicației (preluare imagine, procesare „mirror”, procesare Sobel) este măsurată pentru a evalua performanța aplicației. Timpul de execuție pentru fiecare operațiune este calculat și afişat pe pagina web într-un container de loguri. Aceasta permite utilizatorilor să monitorizeze performanța aplicației în timp real.
6. **Scrierea pe canvas cu întârziere:** După procesarea imaginii, aceasta este scrisă pe canvas cu o întârziere deliberată setată prin funcția `setTimeout()`, pentru a permite imaginii să se încarce complet înainte de a fi redată pe pagina web.

Descrierea modulelor

- **fetchRandomDogImage():** Acest modul este responsabil pentru preluarea imaginii aleatorii de la API-ul DogCEO. Utilizând funcția `fetch()`, se solicită un URL al unei imagini de tip „random breed” din API. Imaginea este apoi încărcată folosind funcția `loadImage()`, iar după ce este încărcată complet, aceasta este afișată pe canvas.
- **loadImage():** Funcția `loadImage()` gestionează procesul de încărcare a imaginii într-un obiect `Image`. Aceasta returnează o promisiune care se rezolvă când

imaginea este încărcată complet sau se respinge în cazul unui eșec de încărcare.

- **displayImage():** Afișează imaginea pe canvas. Imaginea este scalată corect și poziționată în centru, pentru a respecta dimensiunile originale ale imaginii.
-
- **mirrorImage():** Aplică un efect de reflexie pe axa verticală. Aceasta procesează imaginea pixel cu pixel, inversând ordinea pixelilor pe fiecare linie a imaginii pentru a crea un efect de simetrie. Timpul de execuție al acestui proces este măsurat și înregistrat pentru performanță.
- **SobelOperator():** Aplică operatorul Sobel pentru detectarea contururilor. Imaginea este convertită mai întâi într-o imagine în tonuri de gri, iar apoi se aplică operatorii Sobel pe direcțiile orizontală și verticală pentru a calcula gradienții și a obține magnitudinea contururilor. Această tehnică este foarte utilă pentru extragerea informațiilor relevante dintr-o imagine.

- **processImageInSlices():** Imparte imaginea într-un număr de patru secțiuni și aplică operatorul Sobel pentru detectarea contururilor în fiecare secțiune, procesându-le asincron. În timpul procesării, imaginea este tratată într-un canavas temporar pentru a evita actualizările directe ale imaginii pe ecran, iar rezultatul este combinat și actualizat doar în regiunile care nu se suprapun. De asemenea, între procesarea fiecărei secțiuni, există o pauză de 1 secundă pentru a îmbunătăți performanța percepută.

  Timpul de execuție este măsurat pentru a evalua performanța procesării pe secțiuni.

- **processImageInSlices2():** procesează o imagine în patru secțiuni (slices) pentru a aplica un efect de reflexie asupra fiecărei secțiuni într-un mod asincron, cu o întârziere de 1 secundă între fiecare secțiune procesată.

  În primul rând, se obține imaginea de pe canvas și se creează un canvas temporar pentru manipularea datelor imaginii. Apoi, imaginea este împărțită în patru secțiuni orizontale, iar pentru fiecare secțiune se aplică efectul de reflexie. Aceasta se face prin schimbarea pixelilor din partea stângă cu cei din partea dreaptă a secțiunii respective. După procesare, secțiunea este reintrodusă pe canvasul temporar și actualizată pe canvasul principal.

- **logExecutionTime():** Măsoară timpul de execuție al fiecărei operațiuni și înregistrează aceste informații într-un container de loguri pe pagina web, astfel încât utilizatorii să poată urmări performanța aplicației pe parcursul utilizării acesteia.

Bibliografie

- **MDN Web Docs – Canvas API:** Documentație detaliată despre API-ul Canvas din JavaScript (https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API).
- **Dog CEO API Documentation:** Documentația oficială pentru API-ul DogCEO, care oferă imagini aleatorii cu câini din diverse rase (https://dog.ceo/dog-api/).
- **Sobel Operator – Wikipedia:** Explicații teoretice și detalii despre operatorul Sobel și utilizările sale în procesarea imaginilor (https://en.wikipedia.org/wiki/Sobel_operator).