Monica Pascu
CTI – Grupa 351

Proiect 2

Redimensionarea imaginilor cu păstrarea conținutului

Există mai multe metode de a redimensiona o imagine , precum tăierea (cropping) sau scalarea (scaling) cu un factor. Prin scalare se reduc detaliile importante , iar tăierea alterează conținutul imaginii. În acest proiect se abordează o altă metodă , propusă de S.Avidan și A.Shamir în articolul lor "Seam-Carving for Content-Aware Image Resizing".

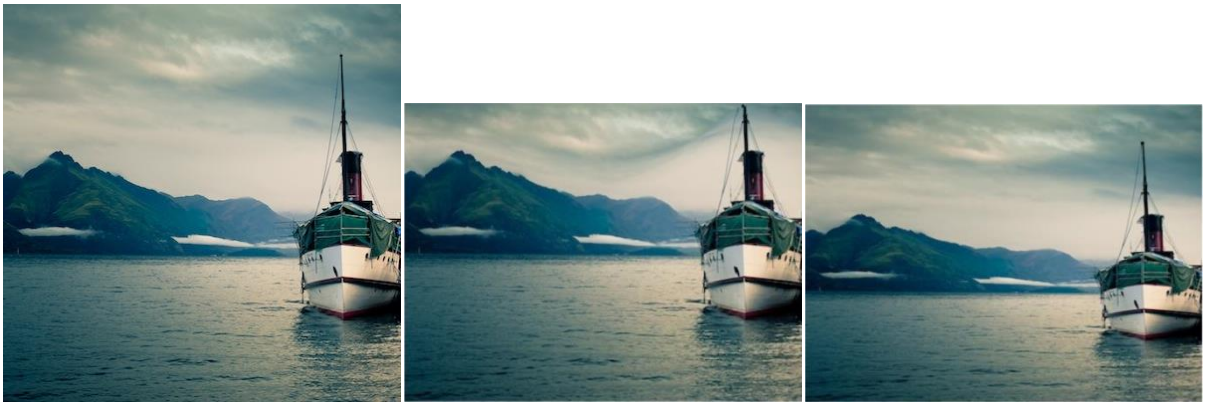
Algoritmul constă în a modifica dimensiunile imaginii prin eliminarea/adăugarea "cusăturii" (drum de pixeli) celei mai nesemnificative din imagine. Spre exemplu , o cusătură verticală este un drum de pixeli alăturați care pornește din vârful imaginii până la baza acesteia. Pentru ca algoritmul să fie eficient , suma energiilor pixelilor ce alcătuiesc drumul trebuie să fie minimă , în sensul că nu există un alt drum de o sumă mai mică.

Proiectul abordează 3 metode de selectare a drumurilor : aleatoriu , greedy și programare dinamică. Cea mai eficientă este metoda programării dinamice.

(a) Micșorarea imaginii pe lățime

imaginea originală imaginea rezultată imresize

(b) Micșorează înălțimea



(c) Mărirea imaginilor

Pentru a mări imaginile , am calculat primele k drumuri (unde k reprezintă numărul de pixeli cu care vrem să mărim imaginea) iar apoi le-am dublat. Nu am abordat , însă , cazul în care 2 sau mai multe drumuri se intersectează.

Imagine originală imaginea rezultată imresize

(d) Amplificarea conținutului imaginilor

Pentru acest subpunct am scalat imaginea inițială cu un factor de 1.2 (factorul poate fi schimbat din funcția *ruleazaProiect.m*) iar apoi am micșorat imaginea pe lungime și înălțime până am ajuns la mărimea inițială.

imagine originală

imaginea rezultată

(e) Eliminarea unui obiect din imagine

Pentru a elimina un obiect din imagine am folosit funcția *ginput.m* pentru a selecta obiectul de eliminat , iar apoi , folosind funcția *poly2mask.m* , am creat o mască pe care am aplicat-o matricei de energie (E) , modificând pixelii obiectului astfel încât să aibă valoarea –inf (pentru ca drumurile să treacă sigur prin el).

(f) Rezultate obținute pentru alte imagini (minim 5 , dintre care minim 2 eșecuri)

1 – Amplificarea conținutului

imaginea originală

imaginea rezultată folosind metoda aleatoare

programare dinamică

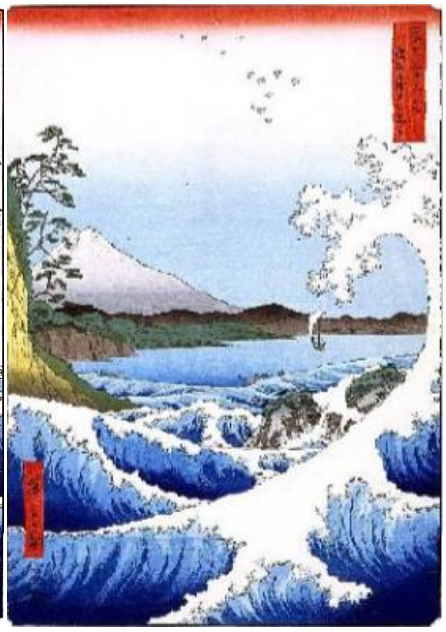
greedy

imresize

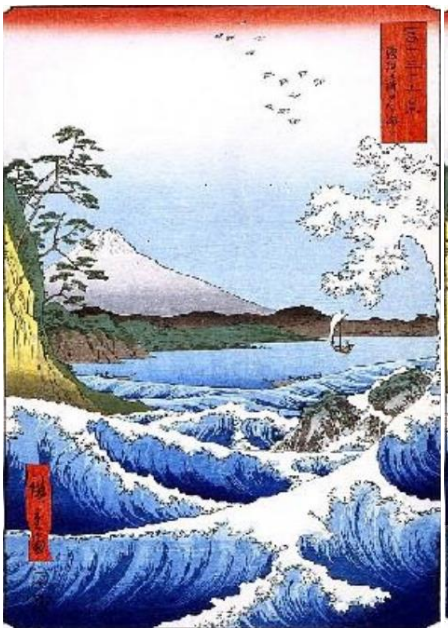
Algoritmul funcționeză bine datorită conținutul imaginii.

2 – Micșorare pe lățime cu 200 pixeli

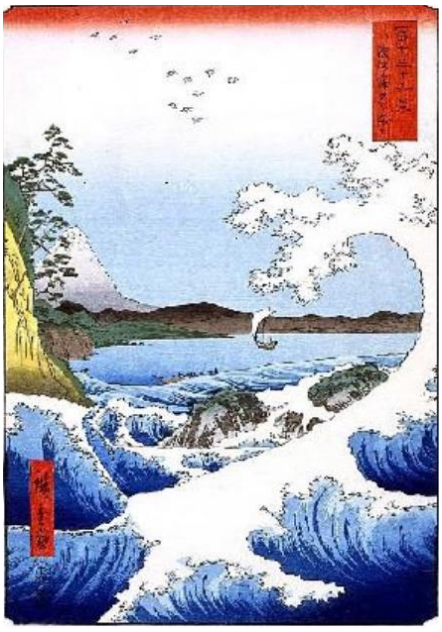
imaginea originală



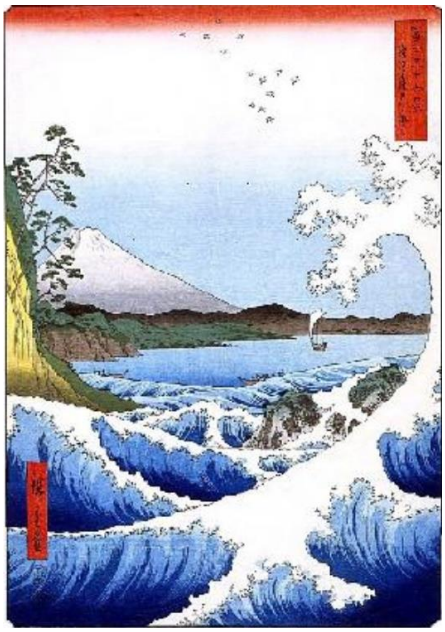
aleatoriu



programare dinamică



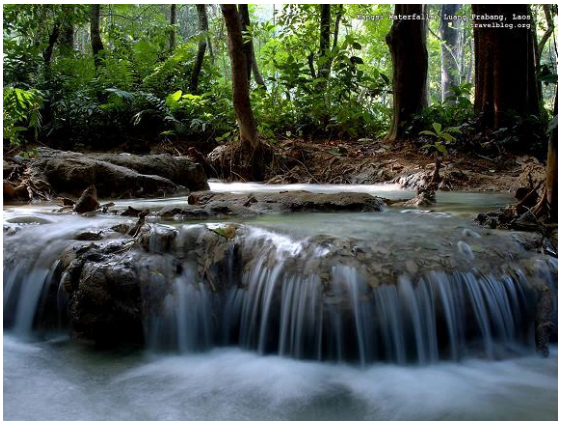
greedy



imresize

Putem să ne dăm seama că algoritmul funcționează bine întrucât muntele Fuji , stâncile , stolul de păsări și semnătura din dreapta-sus au rămas aproape intacte. Cele două stânci s-au apropiat una de alta , însă abia și-au schimbat mărimea.

3 – Micșorarea pe lungime cu 100 pixeli

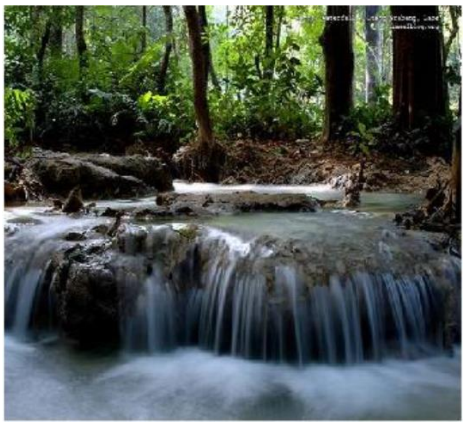


imaginea originală

aleatoriu

programare dinamică

greedy



imresize

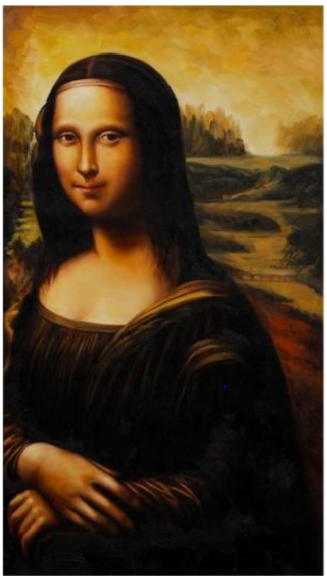
În acest exemplu algoritmul funcționează foarte bine deoarece , într-o imagine cu atât de multe detalii , eliminarea celor mai ”nedetaliate” drumuri nu este atât de observabilă . Sigur , copacii sunt mult mai subțiri , dar dacă persoana care se uită la imagine nu cunoaște originalul , nu își va da seama.

4 – Micșorarea pe lățime cu 100 de pixeli

imaginea originală

aleatoriu

programare dinamică



greedy

imresize

Ochiul uman este sensibil la simetria si proporțiile unui obiect. Algoritmul , însă , nu este. Pus în situația de a micșora o imagine cu multe forme , nu are succes.

5 – Micșorarea pe lățime cu 200 de pixeli

imaginea originală

aleatoriu

programare dinamică

greedy

imresize

Atât abordarea greedy cât si cea aleatoare prezintă mai mult succes decât programarea dinamică. În acest caz , părțile cele mai importante din imagine au mai puțină energie decât restul , astfel ce vrem să rămână intact este , din nefericire, șters.