Redimensionarea imaginilor cu păstrarea conținutului

Hagiu Bogdan 353

În acest proiect s-au realizat funcții pentru micșorarea, mărirea, amplificarea și eliminarea conținutului dintr-o imagine. În funcția ”*selecteazaDrumVertical*” găsim trei metode de parcurgere a imaginilor.

A. Aleator

Modul aleator presupune alegerea unui pixel arbitrar în mod aleator de pe prima linie a imaginii și coborârea pe coloane pe cele trei direcții posibile: diagonală stânga, jos, diagonală dreapta. Pentru a nu exista erori s-a ținut cont de modul de coborâre la marginile imaginii.

```
case 'aleator'
    %pentru linia 1 alegem primul pixel in mod aleator
    linia = 1;
    %coloana o alegem intre 1 si size(E,2)
    coloana = randi(size(E,2));
    %punem in d linia si coloana coresponzatoare pixelului
    d(1,:) = [linia coloana];
    for i = 2:size(d,1)
        %alege urmatorul pixel pe baza vecinilor
        %linia este i
        linia = i;
        %coloana depinde de coloana pixelului anterior
        if d(i-1,2) == 1%pixelul este localizat la marginea din stanga
            %doua optiuni
            optiune = randi(2)-1;%genereaza 0 sau 1 cu probabilitati egale
        elseif d(i-1,2) == size(E,2)%pixelul este la marginea din dreapta
            %doua optiuni
            optiune = randi(2) - 2; %genereaza -1 sau 0
        else
            optiune = randi(3)-2; % genereaza -1, 0 sau 1
        end
        coloana = d(i-1,2) + optiune;%adun -1 sau 0 sau 1:
        % merg la stanga, dreapta sau stau pe loc
        d(i,:) = [linia coloana];
    end
```

B. Greedy

Metoda Greedy este asemănătoare cu modul aleator, diferența fiind dată de criteriul de selecție al următorului pixel. Folosindu-se de funcția ”*calculeazaEnergie*”, metoda Greedy alege pixelul învecinat care are valoarea energiei cea mai mică.

```
case 'greedy'
    %completati aici codul vostru
    linia = 1;
    [~,coloana] = min(E(1,:));
    d(1,:) = [linia coloana];
    for linia = 2:size(E,1)
        if coloana == 1
            [~,optiune] = min(E(linia,coloana:coloana+1));
            optiune = optiune - 1;
        elseif coloana == size(E,2)
            [~,optiune] = min(E(linia,coloana-1:coloana));
            optiune = optiune - 2;
        else
            [~,optiune] = min(E(linia,coloana-1:coloana+1));
            optiune = optiune - 2;
        end

        coloana = d(linia-1,2) + optiune;
        d(linia,:) = [linia coloana];
    end
```

C.Programare dinamică

Metoda presupune realizara unei matrice de energii de dimensiunea imaginii folosite și parcurgerea acesteia de jos în sus, selectând de fiecare dată pixelul de valoare energetică minimă. Astfel se realizează drumul de cost minim ce urmează a fii prelucrat.

```
case 'programareDinamica'
    %completati aici codul vostru
    M = zeros(size(E));
    M = double(M);
    M(1,:) = E(1,:);
    for i  = 2:size(E,1)
        linia = i-1;
        for coloana = 1:size(E,2)
            if coloana == 1
                [val,~] = min([M(linia,coloana),M(linia,coloana+1)]);
                M(i,coloana) = val + E(i,coloana);
            elseif coloana == size(E,2)
                [val,~] = min([M(linia,coloana-1),M(linia,coloana)]);
                M(i,coloana) = val + E(i,coloana);

            else
                [val,~] = min([M(linia,coloana-1),M(linia,coloana),M(linia,coloana+1)]);
                M(i,coloana) = val + E(i,coloana);
            end
        end
    end
    liniaL = size(M,1);
    [~,coloanaL] = min(M(liniaL,:));
    d(liniaL,:) = [liniaL coloanaL];
    for i =liniaL-1:-1:1
        if coloanaL == 1
            [~,poz] = min([M(i,coloanaL),M(i,coloanaL+1)]);
            poz = poz -1;

        elseif coloanaL == size(M,2)
            [~,poz] = min([M(i,coloanaL-1),M(i,coloanaL)]);
            poz = poz - 2;
        else
            [~,poz] = min([M(i,coloanaL-1),M(i,coloanaL),M(i,coloanaL+1)]);
            poz = poz - 2;
        end
        coloanaL = d(i+1,2) + poz;
        d(i,:) = [i coloanaL];
    end
```

Calculul Energiei

Calcularea valorii energetice a pixelului se realizează prin aplicarea unui filtru SOBEL pe varianta alb-negru a imaginii folosite.

```
img = rgb2gray(img);
f = fspecial('sobel');
fx = imfilter(double(img), f, 'replicate');
fy = imfilter(double(img), f', 'replicate');
E = abs(fx) + abs(fy);
```

1.1 Micșorarea imaginii pe lățime

Imagine Inițială:

A.Aleator

B. Greedy

C.ProgDinamică

D.Redimensionare clasică

1.2 Micșorarea imaginii pe înălțime

Imagine Inițială:

A.Aleator

B. Greedy

C.ProgDinamică

D.Redimensionare clasică

1.2 Mărirea imaginilor

Imagine Inițială:

A.Aleator

B. Greedy

C.ProgDinamică

D.Redimensionare clasică

1.4 Amplificarea conținutului imaginilor

Imagine Inițială:

A.Aleator

B. Greedy

C.ProgDinamică

D.Redimensionare clasică

1.5 Eliminarea unui obiect/persoană din imagine

Imagine Inițială:

A.Aleator

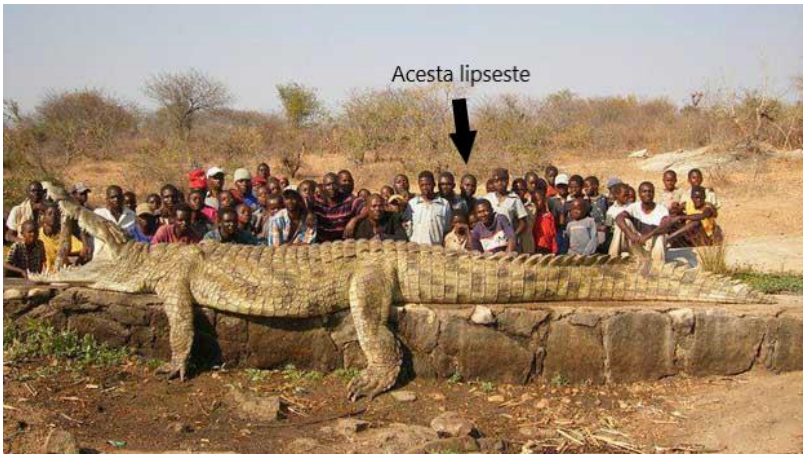
B. Greedy

C.ProgDinamică

D.Redimensionare clasică

1.6.1. Eliminarea unui obiect/persoană

Imagine inițială:



Rezultat:

Algoritmul a funcționat deoarece fiecare drum de cost minim global conține o parte din porțiunea selectată.

1.6.2. Micșorarea imaginii pe lățime

Imagine inițială:

Rezultat:

Algoritmul a funcţionat deoarece am eliminat suficient de puțini pixeli pentru a nu apărea probleme.

1.6.3. Eliminarea unui obiect/persoană

Imagine inițială:

Rezultat:

Algoritmul nu a funcționat deoarece drumurile care trebuie să treacă prin porțiunea selectată, inevitabil trec și prin jucătorii de deasupra.

1.6.4. Mărirea imaginilor

Imagine inițială:

Rezultat:

Algoritmul nu a funcționat deoarece poza aleasă are nivelul energiei aproximativ constant.

1.6.4. Amplificarea conținutului imaginilor

Imagine inițială:

Rezultat:

Algoritmul a funcționat deoarece este un singur obiect/ființă în primplan.

Obs. ***Nu au fost introduse în pdf rezultatele obținute cu metodele aleator și Greedy deoarece nu erau elocvente cerinței (f)!***