MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Metoda SUSAN de detectie a colturilor dintr-o imagine grayscale

Interactiune Om-Calculator

Student: **Muresan Ioana Danina**

2023

Cuprins

Capitolul 1. Introducere

Metoda Susan reprezintă o tehnică în prelucrarea imaginilor, specializată în detectarea colțurilor semnificative. Dezvoltată în anii 1990, această metodă se concentrează pe identificarea variațiilor de intensitate pentru a localiza colțurile într-o imagine. Ea este cunoscută pentru adaptabilitatea sa la diverse condiții de iluminare și rezistența la zgomot în imagini.

Capitolul 2. Obiectivele proiectului

Principalele obiective ale proiectului metodei Susan de detectare a colțurilor sunt:

Înțelegerea fundamentelor metodei Susan

Metoda Susan (Smallest Univalue Segment Assimilating Nucleus) se distinge prin capacitatea sa de a minimiza numărul de raportări false, adică identificarea eronată a colțurilor. Această caracteristică contribuie semnificativ la precizia sa în localizarea punctelor semnificative în imagini.

De asemenea, este proiectată pentru a asigura o localizare precisă a colțurilor în imagini. Acest aspect este esențial în aplicații care depind de exactitatea poziționării punctelor cheie, cum ar fi recunoașterea de obiecte sau navigarea robotică.

O altă caracteristică notabilă a Metodei Susan este independența sa față de variațiile de luminozitate în imagini si de condintiile de zgmot. Această abordare face ca algoritmul să fie robust în diverse condiții de iluminare, asigurând consistență în performanța.

Dezvoltarea unei implementari proprii a metodei Susan

Capitolul 3. Studiu bibliografic

În cadrul documentării bibliografice, am consultat Cursul 4 al materiei ”Interacțiune Om-Calculator”. Informațiile relevante extrase din acest curs pot fi găsite în detaliu în documentul online disponibil la adresa. [1]

Capitolul 4. Analiză și fundamentare teoretică

Principii functionale

- definitia coltului: Un colț este definit ca intersectarea a cel puțin două muchii cu orientări diferite într-un punct.
- punct de interes: Un punct de interes este o poziție bine definită în imagine, detectată robust, și poate fi un colț, minim sau maxim local de intensitate, terminația unei linii sau un punct de pe un contur cu curbura maximă locală.
- robustitate: Metoda SUSAN este robustă, permițând detectarea aceluiași set de colțuri în condiții de iluminare diferite sau transformări geometrice ale imaginii (rotație, scalare, transformare de perspectivă).

Pasii algoritmului USAN (pentru fiecare punct din imagine)

- Plasarea unei măşti circulare in jurul pixelului studiat.
- Calcularea numărului de pixeli din USAN (n).
- Calculul răspunsului iniţial (R) prin scăderea din pragul geometric a valorii USAN-ului.
- Eliminarea raportărilor false de colţuri prin folosirea metodelor ce implica centrul de greutate si continuitatea USAN-ului.
- Aplicare non-maxima suppression -> colturi finale.

Capitolul 5. Proiectare de detaliu și implementare

5.1. Proiectare

- **Kernelul pentru Maska Circulară**
  În cadrul proiectării, un aspect esențial este definirea kernelului pentru masca circulară. Acesta este de dimensiuni 7x7 și este utilizat pentru a extrage o regiune locală în jurul fiecărui pixel din imagine. Kernelul este definit astfel încât să excludă anumite poziții și să obțină forma circulară dorită.

$$Kernel = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & X & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Figura 5.1: Kernel

- **Algoritmul de Detecție SUSAN**
  Algoritmul de detecție a colțurilor SUSAN este implementat în funcția `susancornerdetection`. Acesta parcurge pixel cu pixel imaginea și aplică masca circulară pentru a calcula o valoare de răspuns în funcție de diferența de intensitate dintre pixelul curent și cei din jur. Valoarea de răspuns este apoi comparată cu un prag geometric pentru a determina dacă pixelul curent reprezintă sau nu un colț.

- **Filtrarea Răspunsului Inițial**
  Răspunsul inițial al algoritmului SUSAN este supus filtrării în funcția `filterresponse`. Acest proces implică calculul centrului de masă al USAN-ului (Smallest Univalue Segment Assimilating Nucleus) și compararea distanței de la acesta la centrul nucleului. Se verifică, de asemenea, continuitatea USAN-ului pentru a elimina potențialele colțuri false.

- **Suprimarea Non-Maximelor**
  Ultimul pas al algoritmului constă în suprimarea non-maximelor pentru a obține colțurile finale. Aceasta se realizează în funcția `non-max-suppression` prin compararea valorii fiecărui pixel cu valorile din vecinătate, eliminând astfel pixelii care nu sunt maxime locale.

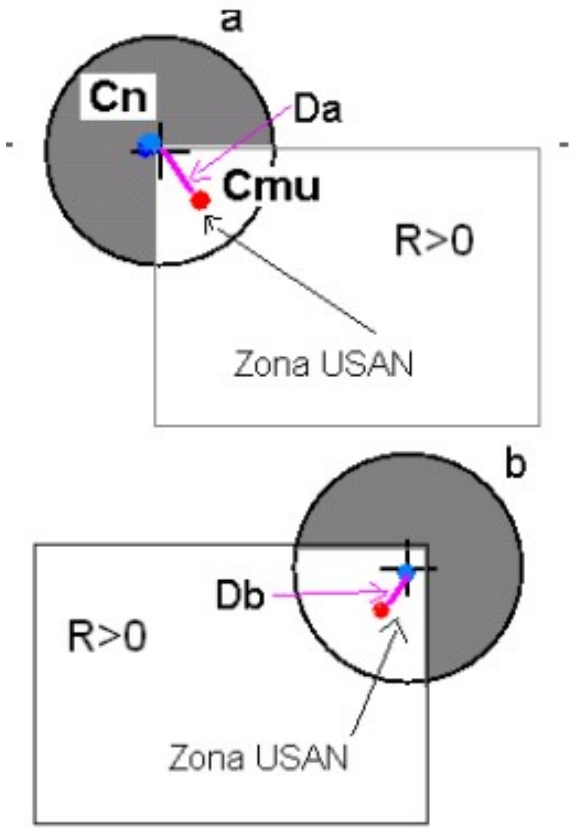

Figura 5.2: Cazuri Valide

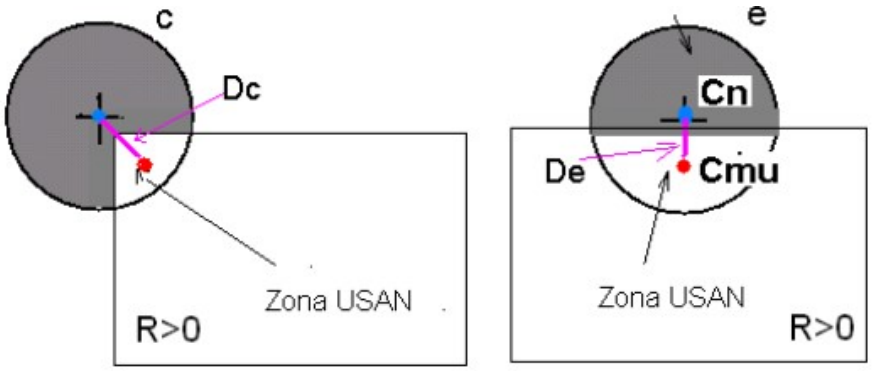

Figura 5.3: Cazuri Invalide

5.2. Implementare

`Susan_corner_detection` este principala functie a proiectului. Aici se inițializează variabila img pentru a asigura că imaginea este reprezentată ca un tablou de numere reale (float64). Aceasta îmbunătățește precizia în calculele ulterioare.Se definește pragul geometric g, iar măscarea circulară este obținută utilizând funcția `kernel_mask`.O matrice goală de dimensiuni egale cu dimensiunile imaginii este inițializată pentru a stoca rezultatele algoritmului.Algoritmul parcurge imaginea pixel cu pixel, excluzând o margine de 3 pixeli pentru a asigura aplicarea măștii în jurul fiecărui pixel.Se extrage o regiune de 7x7 din jurul fiecărui pixel, iar măscarea circulară este aplicată pentru a selecta doar anumite poziții.Se calculează răspunsul SUSAN (a) pentru pixelul curent folosind formula specifică algoritmului.Răspunsul SUSAN este comparat cu pragul geometric g, iar rezultatul este stocat în matricea de ieșire.Funcția returnează matricea de ieșire, care conține răspunsurile pentru fiecare pixel.Această funcție efectuează operațiile de bază ale algoritmului SUSAN pentru a detecta colțurile în imaginea de intrare. Rezultatul poate fi apoi utilizat pentru a continua procesul de filtrare și suprimare a non-maximelor.

Functia `filter_response(output)` se ocupă de filtrarea răspunsului inițial al algoritmului SUSAN pentru a elimina colțurile false și pentru a păstra doar colțurile valide. Se inițializează matricele Cmu și Cn cu valori zero. Acestea vor fi folosite pentru a calcula centrul de masă al USAN-ului (Cmu) și distanța D de la centrul de masă la centrul nucleului (Cn). Variabila a este setată pentru a ajusta comportamentul algoritmului.Se parcurge matricea de răspuns a algoritmului SUSAN și se calculează centrul de masă (Cmu) și distanța D (Cn) pentru fiecare pixel, folosind o fereastră 7x7 și măscă circulară.Se verifică dacă distanța D pentru pixelul curent este maximă în fereastra 7x7. Dacă da, pixelul reprezintă un colț real și nu este eliminat. În caz contrar, se verifică continuitatea USAN-ului prin compararea diferenței dintre distanța D și centrul de masă (Cmu). Dacă toți pixelii de pe segmentul D fac parte din USAN, atunci pixelul curent nu reprezintă un colț valid și este eliminat (setat la 0 în matricea de ieșire).

Functia `non_max_suppression(output,neighborhood-size)` primește ca parametri matricea de răspuns a algoritmului SUSAN (output) și dimensiunea vecinătății (neighborhood-size) folosită pentru a determina localul maxim.Variabilele h și w reprezintă înălțimea și lățimea matricei de răspuns. Variabila half-size reprezintă jumătate din dimensiunea vecinătății și este folosită pentru a accesa submatricea în jurul fiecărui pixel.Se parcurge matricea de răspuns, excluzând o margine de dimensiunea vecinătății pentru a asigura că algoritmul nu verifică marginile imaginii, unde nu pot exista colțuri.Se calculează valoarea maximă din vecinătatea definită în jurul pixelului curent.Dacă valoarea pixelului curent (output[i, j]) este mai mică decât valoarea maximă locală (local-max), atunci pixelul curent nu este un maxim local și este suprimat (setat la 0).Funcția returnează matricea de răspuns actualizată după etapa de suprimare a non-maximelor.Această etapă are rolul de a elimina redundanțele în răspunsul algoritmului, astfel încât să rămână doar colțurile finale, cele care reprezintă maxime locale în vecinătatea lor.

Capitolul 6. Testare și validare

Pentru testare, am folosit mai mult imagini care contin colturi. Am afisat in paralel imaginea obtinuta prin metoda Susan fara a avea eliminate colturile false, fara rafinare si imaginea finala obtinuta dupa aplicare non-maxima suppression. Se observa clar o imbunatatire, numarul colturilor au fost reduse semnificativ.

Metoda implementata de mine NU functioneaza perfect, pentru unele imagini de test nu se comporta asa cum ar trebui. De exemplu, pentru imaginea cu tabla de sah afiseaza mai multe colturi decat ar trebui, dar se vede clar ca adevaratele colturi sunt mai ingrosate. Pentru a corecta acest lucru, am tot facut debugging, am modificat parametrii(tresholdul, mask etc) dar fara vreun rezultat.

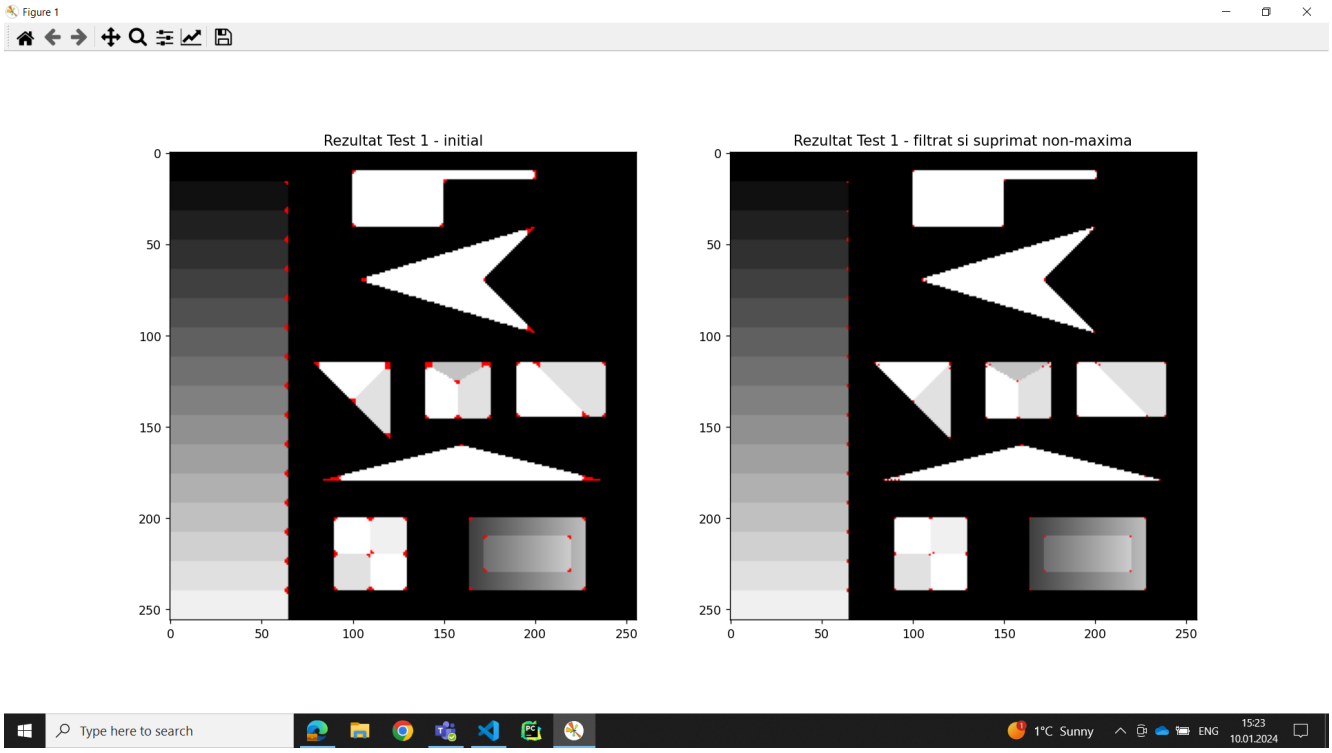

Figura 6.1: Functioneaza cum se asteapta

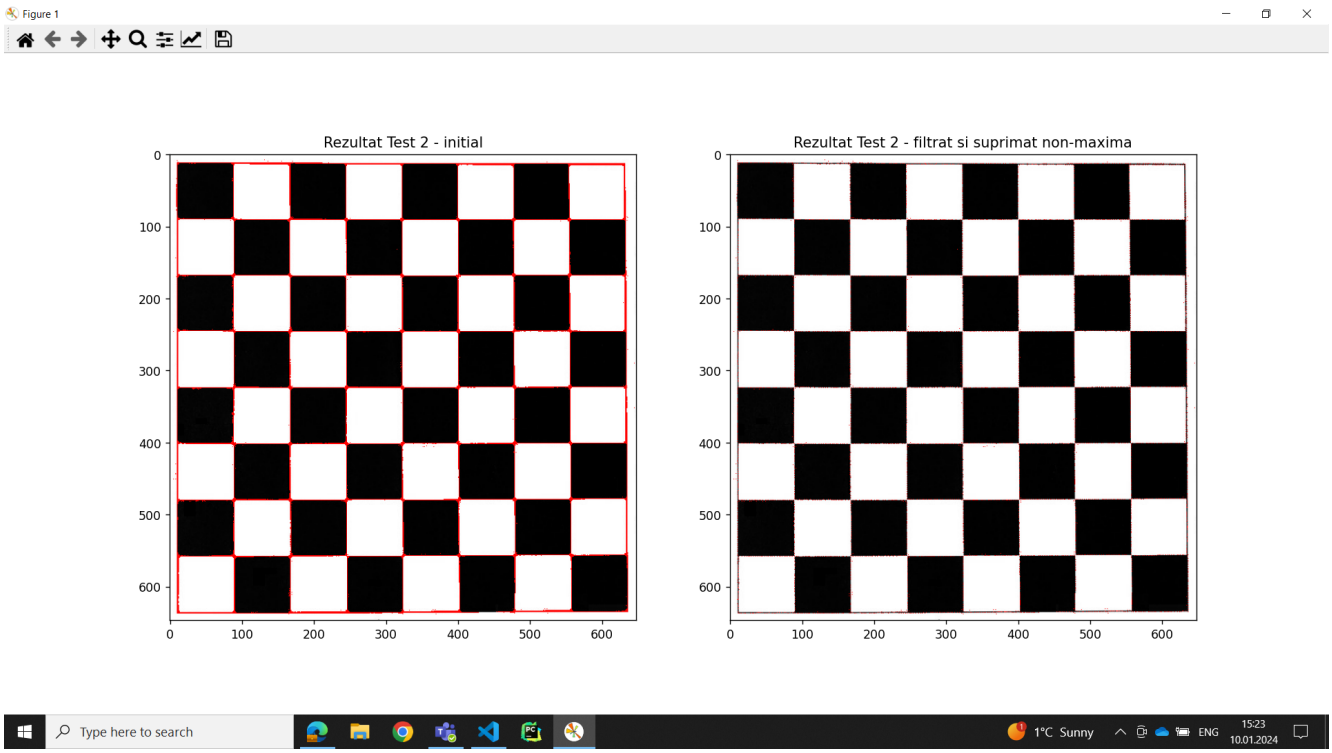

Figura 6.2: NU functioneaza cum se asteapta

Capitolul 7. Manual de instalare și utilizare

Pentru a testa programul, este suficient sa fie instalat un IDE pe laptop. Eu am folosit Visual Studio Code. Si rulez programul utilizand comanda intr-un terminal **python susanCornerDetection.py**.

Imaginile se iau automat din folderul proiectului. Daca se doreste testarea si pe alte imagini, acestea trebuie adaugate in folder si creata un nou test pentru aceasta imagine, sau sa se modifice un test deja existent cu noua imagine.

```
112  #primul test - rezultatul initial
113  img1 = cv.imread("susan_input1.png", 0)
114  output1 = susan_corner_detection(img1)
115  finaloutput1 = cv.cvtColor(img1, cv.COLOR_GRAY2RGB)
116  finaloutput1[output1 != 0] = [255, 0, 0]
117
118  #primul test - rezultatul filtrat si suprimat non-maxima
119  img2 = cv.imread("susan_input1.png", 0)
120  output2 = susan_corner_detection(img2)
121  output2_filtered = filter_response(output2)
122  output2_suppressed = non_max_suppression(output2_filtered)
123  finaloutput2 = cv.cvtColor(img2, cv.COLOR_GRAY2RGB)
124  finaloutput2[output2_suppressed != 0] = [255, 0, 0]
125
126  #afisarea rezultatelor
127  plot_images_side_by_side(finaloutput1, "Rezultat Test 1 - initial", finaloutput2, "Rezultat Test 1 - filtrat si suprimat non-maxima")
```

Capitolul 8. Concluzii

In concluzie, proiectul implementează metoda SUSAN (Smallest Univalue Segment Assimilating Nucleus) pentru detecția de colțuri în imagini. Algoritmul implementat parcurge pixel cu pixel imaginea, aplicând o mască circulară pentru a calcula un răspuns inițial. Acest răspuns este filtrat și suprimat pentru a elimina colțurile false. Testele au evidențiat îmbunătățiri semnificative, cu toate că există situații în care algoritmul poate prezenta neconcordanțe, cum ar fi detectarea excesivă a colțurilor în anumite condiții. Deși implementarea are anumite limitări, cum ar fi comportamentul nedorit pe imagini specifice (de exemplu, tabla de șah), proiectul reprezintă o abordare eficientă și robustă pentru detecția de colțuri. Dezvoltarea ulterioară ar putea implica optimizări ale parametrilor algoritmului și explorarea altor metode de detectare a colțurilor pentru a obține rezultate mai precise, dar in primul rand sa fie perfect functionabil.

Contributia mea la acest proiect a fost sa ma documentez, sa inteleg metoda si sa incerc sa o aplic cat mai bine, astfel ca teoria de la curs a fost pusa in practica. De asemenea, am testat mai multe metode pentru filtrarea raspunsului pentru a obtine una cat mai buna.

Bibliografie

[1] T. Marita. Interacțiune om-calculator - cursul 4. Universitatea Tehnică din Cluj-Napoca. [Online]. Available: https://users.utcluj.ro/~tmarita/HCI/C4.pdf

Anexa A. Secțiuni relevante din cod

```
def susan_corner_detection(img):

    img = img.astype(np.float64)
    circularMask = kernel_mask()
    output=np.zeros(img.shape)
    val=np.ones((7,7))

    for i in range(3,img.shape[0]-3):
        for j in range(3,img.shape[1]-3):
            ir=np.array(img[i-3:i+4, j-3:j+4])
            ir =  ir[circularMask==1]
            ir0 = img[i,j]
            a=np.sum(np.exp(-((ir-ir0)/10)**6))
            if a<=g:
                a=g-a
            else:
                a=0
            output[i,j]=a
    return output
```