

**Arpad GELLERT
Remus BRAD**

Procesarea Imaginilor
Aplicații

Cuprins

1. Transformări simple ale imaginilor	3
2. Ajustarea contrastului	5
3. Egalizarea histogramei.....	7
4. Operații geometrice.....	9
5. Îndepărtarea zgomotului	13
6. Accentuarea contururilor	15
7. Detecția contururilor	17
8. Detecția de contur bazată pe filtrul Gabor	19
9. Segmentarea imaginilor	20
10. Corelația imaginilor	22
11. Algoritmi <i>block matching</i>	24
Bibliografie	26

1. Transformări simple ale imaginilor

1.1. Transformarea imaginilor din RGB în nivele de gri

Nivelul de gri corespunzător culorii unui pixel se obține prin aducerea la aceeași intensitate a celor trei componente ale culorii pixelului respectiv (R-roșu, G-verde, B-albastru). Procedeu se aplică pentru toți pixelii din imagine. Figura următoare prezintă cele 256 de nivele de gri reprezentate pe 24 biți/pixel:

Negru			Alb
0, 0, 0	1, 1, 1	• • •	255, 255, 255

Figura 1. Nivelele de gri

Intensitatea comună ale celor trei componente RGB, poate fi obținută prin următoarea regulă:

$$I = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

1.2. Negativarea imaginilor

Negativarea unei imagini poate fi efectuată transformând componentele culorii fiecărui pixel folosind următoarea regulă:

$$\begin{cases} R = 255 - R \\ G = 255 - G \\ B = 255 - B \end{cases}$$

1.3. Modificarea luminozității imaginilor

Luminozitatea unei imagini se poate modifica prin adunarea unei valori (Δ) la componentele culorii fiecărui pixel din imaginea respectivă, având însă grijă să nu se iasă în afara intervalului $[0, 255]$. Folosind o valoare pozitivă ($\Delta > 0$), se obține o luminozitate mai deschisă, în timp ce o valoare negativă ($\Delta < 0$) determină o luminozitate mai închisă a imaginii. Modificarea luminozității poate fi efectuată folosind următoarea regulă:

$$R = \begin{cases} 255, & \text{dacă } R + \Delta > 255 \\ 0, & \text{dacă } R + \Delta < 0 \\ R + \Delta, & \text{în rest} \end{cases}$$

$$G = \begin{cases} 255, & \text{dacă } G + \Delta > 255 \\ 0, & \text{dacă } G + \Delta < 0 \\ G + \Delta, & \text{în rest} \end{cases}$$

$$B = \begin{cases} 255, & \text{dacă } B + \Delta > 255 \\ 0, & \text{dacă } B + \Delta < 0 \\ B + \Delta, & \text{în rest} \end{cases}$$

Aplicații

Să se implementeze tehnicile prezentate în lucrare. Pentru modificarea luminozității să se folosească un slider prin care să se poată seta nivelul dorit de luminozitate.

2. Ajustarea contrastului

Histograma unei imagini reprezintă numărul de pixeli (frecvența) pentru fiecare intensitate din imaginea respectivă. De exemplu, în figura următoare 84 de pixeli din imagine au intensitatea 5:

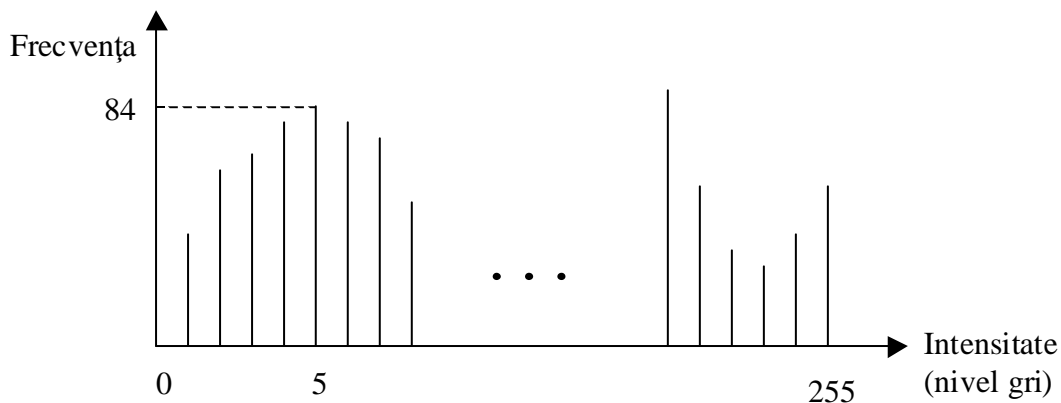


Figura 2. Histograma

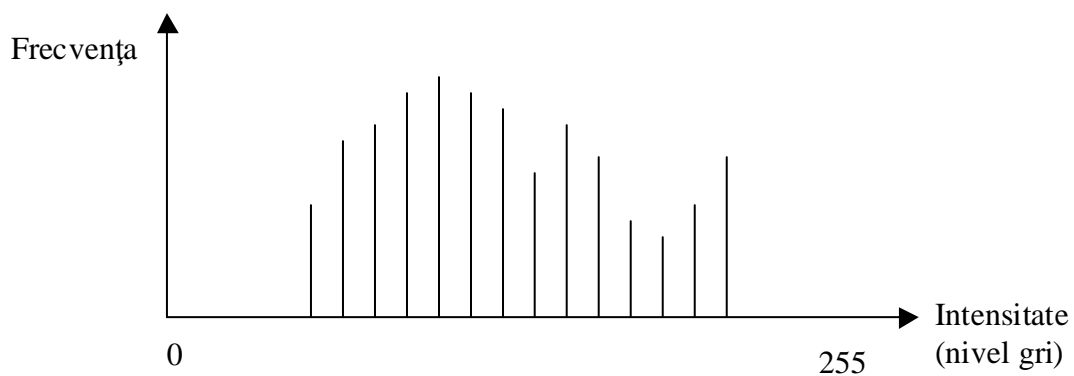


Figura 3. Histograma înainte de ajustarea contrastului

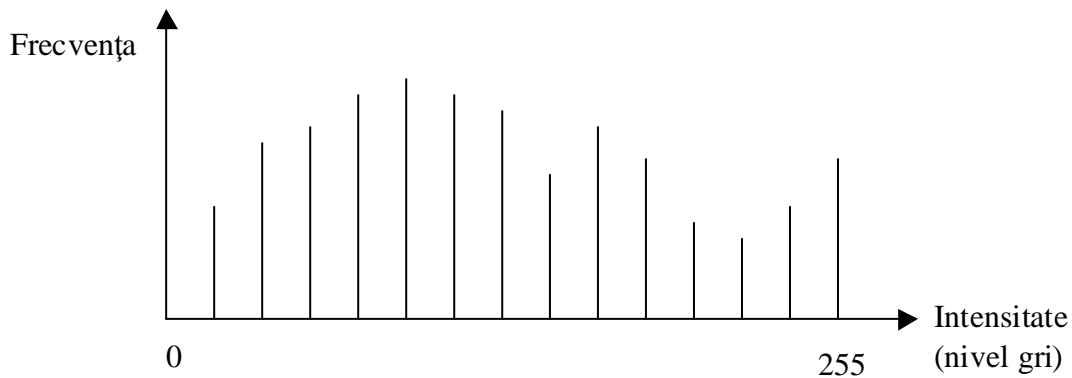


Figura 4. Histograma după ajustarea contrastului

Ajustarea contrastului unei imagini poate fi realizată prin rescalarea intensității fiecărui pixel folosind formula de mai jos:

$$I_N = \frac{b - a}{\max - \min} \cdot (I_O - \min) + a$$

- I_N – noua intensitate;
- I_O – intensitatea originală;
- min – intensitatea minimă în imagine (cea mai închisă);
- max – intensitatea maximă în imagine (cea mai deschisă);
- (max-min) – intervalul original de intensități;
- a – noua intensitate minimă;
- b – noua intensitate maximă;
- (b-a) – noul interval de intensități.

Aplicații

Să se implementeze algoritmul de ajustare a contrastului prin metoda prezentată.

3. Egalizarea histogramei

Egalizarea histogramei este o metodă de modificare neadaptivă a histogramei imaginilor și are rolul de a scoate în evidență informații care pot fi greu identificate în imaginea originală. Etapele algoritmului de rescalare sunt următoarele:

1. Se determină histograma imaginii;
2. Se construiește histograma cumulativă;
3. Se determină vectorul de transformare a histogramei;
4. Se modifică imaginea conform vectorului de transformare.

Efectul procesului de egalizare a histogramei poate fi observat în figura următoare:

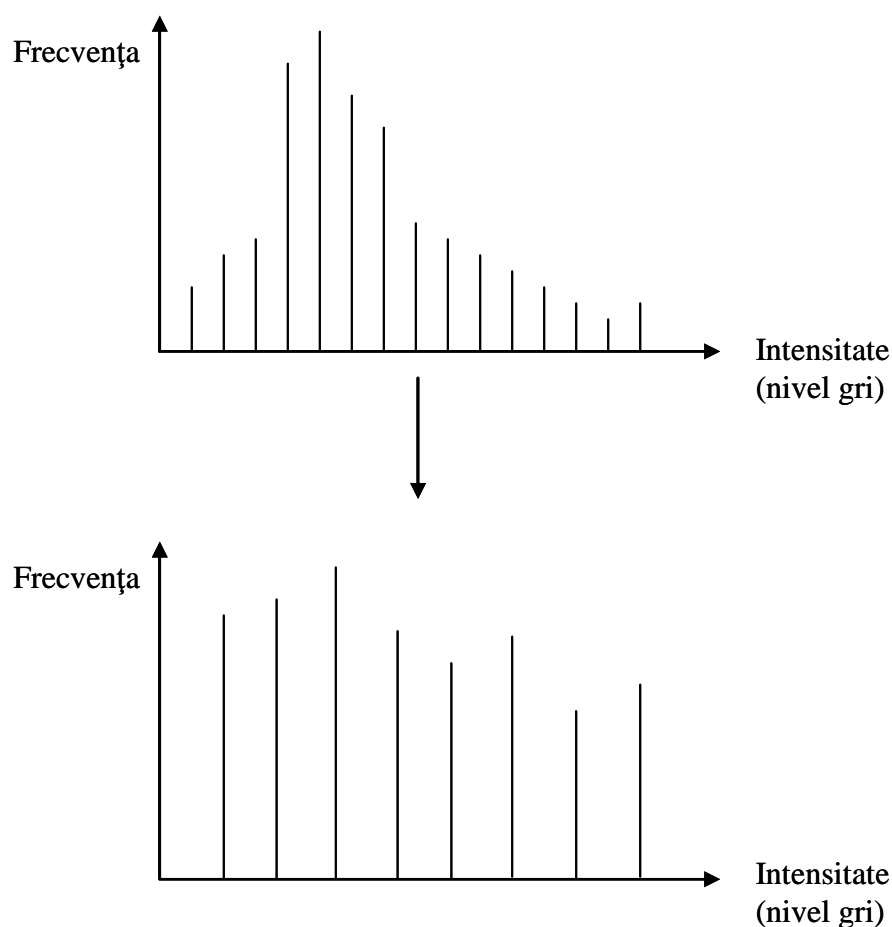


Figura 5. Egalizarea histogramei

Dacă numărul de niveluri de rescalare a imaginii originale e mare, e posibilă modificarea nivelelor de gri în așa fel încât histograma imaginii îmbunătățite să fie aproape constantă.

Algoritmul de egalizare a histogramei este următorul:

```
for r ← 1 to Height do
  for c ← 1 to Width do
    i ← geti(r, c)
    hist[i] ← hist[i] + 1
```

```

for i ← 1 to 255 do
    histc[i] ← histc[i-1] + hist[i]

for i ← 0 to 255 do
    transf[i] = (histc[i] * 255) / (Width * Height)

for r ← 1 to Height do
    for c ← 1 to Width do
        i ← geti(r, c)
        seti(transf[i], r, c)

```

unde *hist* este histograma imaginii originale, *histc* este histograma cumulativa, *transf* este vectorul care păstrează modificările intensităților, *geti* și *seti* preia respectiv setează intensitatea pixelului de pe o anumită poziție din imagine.

Aplicație

Să se implementeze algoritmul de egalizare a histogramei.

4. Operații geometrice

Transformările geometrice preiau informațiile despre pixelii din imaginea sursă și le mapează, în noile locații din imaginea destinație. Forma generală a unei transformări geometrice poate fi exprimată matricial astfel:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = A \times \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + B$$

unde (x_2, y_2) reprezintă coordonatele pixelului din imaginea destinație, (x_1, y_1) reprezintă coordonatele pixelului din imaginea sursă, iar A și B sunt matricile prin care se particularizează transformarea dorită. Translația poate fi implementată prin specificarea valorilor pentru matricea B , în timp ce scalarea, rotația și reflexia pot fi obținute prin setarea matricii A (vezi paragraful 4.5). Pentru corectarea distorsiunilor geometrice introduse în imagini din cauza unor iregularități de perspectivă, se combină aceste transformări elementare, utilizând ambele matrici.

4.1. Scalarea

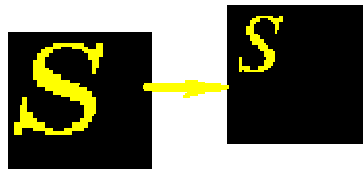


Figura 6. Scalarea

Scalarea este o operație geometrică care mărește sau micșorează o imagine sau o parte a unei imagini. Micșorarea imaginii se realizează prin înlocuirea unui grup de pixeli din imaginea originală (mărimea grupului este determinată de factorul de scalare), cu un singur pixel în imaginea destinație. Acest pixel va avea intensitatea unui anumit pixel din grup, sau intensitatea obținută prin *interpolarea* intensităților pixelilor din grup (de exemplu intensitatea medie).

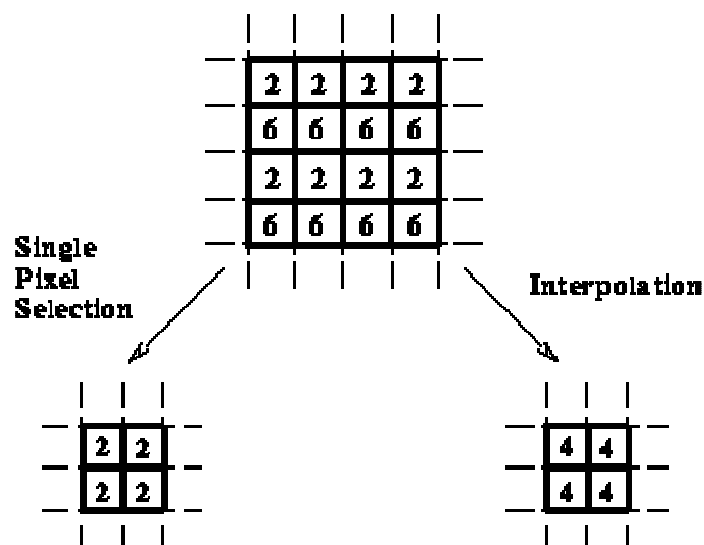


Figura 7. Metode de micșorare. a) Înlocuirea cu pixelul din dreapta sus. b) Interpolarea folosind valoarea medie.

Mărirea unei imagini este realizată prin replicare sau prin interpolare. Astfel, unui pixel din imaginea originală îi corespunde un grup de pixeli în imaginea destinație, care vor avea fie intensitatea pixelului original (din imaginea sursă), fie o interpolare a acesteia cu intensitățile pixelilor învecinați. Mărimea grupului este determinată de factorul de scalare. Operațiile de interpolare, deși sunt mai mari consumatoare de resurse de procesare (ceea ce se traduce prin creșterea timpului de procesare), oferă rezultate superioare din punct de vedere calitativ.

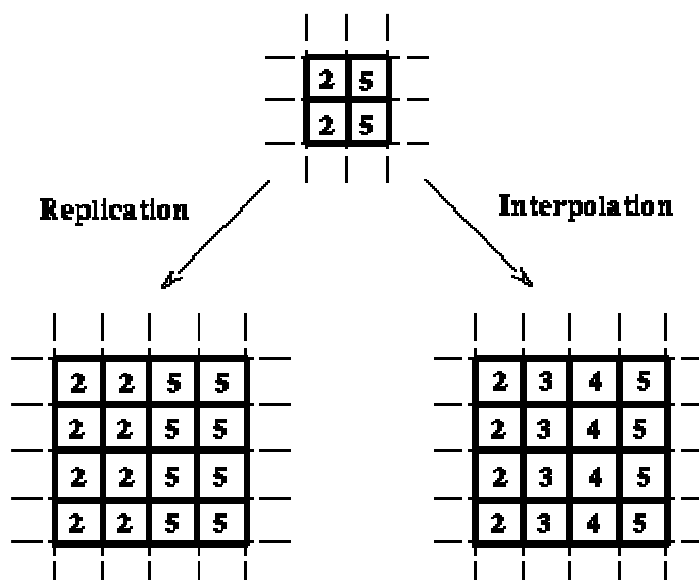


Figura 8. Metode de mărire. a) Replicarea unui singur pixel. b) Interpolarea.

4.2. Rotația

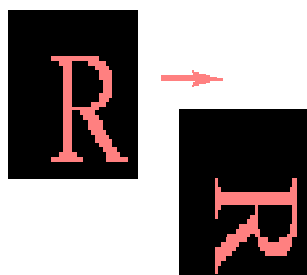


Figura 9. Rotația

Rotația unei imagini se realizează prin maparea unui pixel de intrare (x_1, y_1) în poziția de ieșire (x_2, y_2) , determinată în urma rotirii cu un unghi θ , în jurul unei origini (x_0, y_0) . Transformarea aplicată asupra coordonatelor pixelilor este:

$$x_2 = \cos(\theta) \cdot (x_1 - x_0) - \sin(\theta) \cdot (y_1 - y_0) + x_0$$

$$y_2 = \sin(\theta) \cdot (x_1 - x_0) + \cos(\theta) \cdot (y_1 - y_0) + y_0$$

Rotația este folosită ca procedeu de ajustare/corectare a aspectului unei imagini, sau în cadrul operațiilor de procesare ce acționează direcțional. În majoritatea implementărilor coordonatele de ieșire (x_2, y_2) care se află în afara imaginii sunt ignorate. Algoritmul de rotație poate produce coordonate (x_2, y_2) care nu sunt întregi. Pentru generarea intensităților pixelilor cu coordonate întregi, cele mai importante metode folosite sunt următoarele:

- un pixel cu coordonate întregi poate primi intensitatea celui mai apropiat pixel cu coordonate ne-întregi;
- se calculează intensitățile pixelilor cu coordonate întregi făcându-se media intensităților pixelilor cu coordonate ne-întregi. Această metodă produce rezultate mai bune, dar necesită o putere de procesare mai mare.

4.3. Reflexia

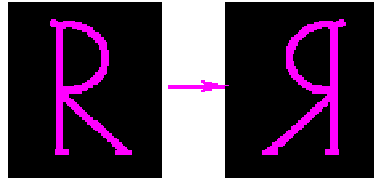


Figura 10. Reflexia

Reflexia geometrică (oglundirea) transformă o imagine sursă astfel încât pixelii (x_1, y_1) sunt reflectați față de o axă specificată în noua poziție (x_2, y_2) în imaginea destinație. Reflexia față de o axă orizontală de ordonată y_0 se realizează prin transformările:

$$\begin{aligned}x_2 &= x_1 \\y_2 &= -y_1 + (2 \cdot y_0)\end{aligned}$$

În mod similar, reflexia față de o axă verticală de abscisă x_0 se realizează astfel:

$$\begin{aligned}x_2 &= -x_1 + (2 \cdot x_0) \\y_2 &= y_1\end{aligned}$$

Reflexia după o axă orientată într-o direcție arbitrara θ , și care trece prin punctul (x_0, y_0) :

$$\begin{aligned}x_2 &= x_1 - 2 \cdot \Delta \cdot \sin(\theta) \\y_2 &= y_1 + 2 \cdot \Delta \cdot \cos(\theta)\end{aligned}$$

unde $\Delta = (x_1 - x_0) \cdot \sin(\theta) - (y_1 - y_0) \cdot \cos(\theta)$. În caz că (x_0, y_0) nu este în centrul imaginii, o parte din imagine va fi reflectată în afara spațiului vizibil al imaginii.

4.4. Translația

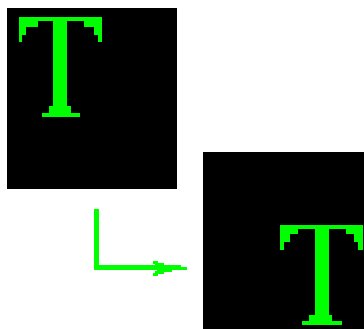


Figura 11. Translația

Translația presupune deplasarea pixelilor (x_1, y_1) din imaginea originală, cu o valoare (β_1, β_2) specificată de utilizator, în noile poziții (x_2, y_2) determinate prin transformările:

$$\begin{aligned}x_2 &= x_1 + \beta_x \\ y_2 &= y_1 + \beta_y\end{aligned}$$

În cazul în care noile coordonate (x_2, y_2) sunt în afara imaginii, operatorul de translație le ignoră.

4.5. Transformarea afină



Figura 12. Transformarea afină

În multe imagini sunt detectate o serie de distorsiuni introduse de iregularitățile de perspectivă. Dacă aceste distorsiuni sunt uniforme, ele pot fi corectate aplicând transformările afine, care mapează pixelii (x_1, y_1) dintr-o imagine de intrare, în noile poziții (x_2, y_2) ale imaginii de ieșire aplicând o combinație liniară a *translației*, *rotației* și a *scalării*. O transformare afină poate fi descrisă prin

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = A \times \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + B$$

Translația pură se obține prin:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Rotația pură folosește doar matricea A :

$$A = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Similar, scalarea pură se obține prin:

$$A = \begin{pmatrix} a_{11} & 0 \\ 0 & a_{22} \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Aplicații

Să se implementeze operațiile geometrice prezentate în lucrare.

5. Îndepărtarea zgomotului

Imaginile pot fi afectate de zgomot cum ar fi zgomotul senzorilor electronici sau zgomotul granulelor fotografice. Pixelii afectați de zgomot au o intensitate diferită de cea a vecinilor lor. Această observație stă la baza multor algoritmi de îndepărtare a zgomotului.

5.1. Filtrul Trece-Jos (FTJ)

Filtrarea trece-jos se poate efectua prin convoluția imaginii afectate de zgomot cu o matrice:

$$H_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad H_2 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad \dots, \quad H_n = \frac{1}{(n+2)^2} \begin{bmatrix} 1 & n & 1 \\ n & n^2 & n \\ 1 & n & 1 \end{bmatrix}.$$

Aceste matrici, numite măști de îndepărtare a zgomotului, sunt normalizate la unitate, astfel procesul de îndepărtare a zgomotului nu introduce o schimbare de amplitudine în imaginea rezultată.

5.2. Tehnica Outlier

O tehnică simplă de îndepărtare a zgomotului, numită *outlier*, compară fiecare pixel cu media a opt dintre vecinii săi. Dacă magnitudinea diferenței e mai mare decât un prag, pixelul este considerat zgomot și e înlocuit cu media pixelilor vecini.

$$\text{dacă } \left| I - \frac{1}{8} \sum_{i=1}^8 p_i \right| > \varepsilon, \text{ atunci } I = \frac{1}{8} \sum_{i=1}^8 p_i.$$

Media poate fi obținută prin convoluția imaginii cu următoarea mască:

$$H = \frac{1}{8} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

5.3. Filtrul median

Filtrul median e format dintr-o fereastră ce cuprinde un număr impar de pixeli. Pixelul central din fereastră e înlocuit cu medianul pixelilor din fereastră:

$$\text{med}(a,b,c,d,e) = \text{maximin}(a,b,c,d,e) = \max \{ \min(a,b,c), \min(a,b,d), \min(a,b,e), \min(a,c,d), \\ \min(a,c,e), \min(a,d,e), \min(b,c,d), \min(b,c,e), \min(b,d,e), \min(c,d,e) \}.$$

5.4. Filtrul pseudomedian

Filtrarea mediană necesită un volum mare de calcule, iar numărul operațiilor crește exponențial cu dimensiunea ferestrei. Un operator mai simplu este filtrul pseudomedian:

$$\begin{aligned}
pmed(a,b,c,d,e) &= \frac{1}{2} \maximin(a,b,c,d,e) + \frac{1}{2} \minimax(a,b,c,d,e) = \\
&= \frac{1}{2} \max\{\min(a,b,c), \min(b,c,d), \min(c,d,e)\} + \frac{1}{2} \min\{\max(a,b,c), \max(b,c,d), \max(c,d,e)\}.
\end{aligned}$$

În cazul unui filtru pseudomedian în formă de “+” calculul funcțiilor *min* și *max* se face în ferestre, după cum urmează:

$$\begin{array}{ccccccc}
& & & & y_1 & & \\
& & & & \vdots & & \\
& & & & \vdots & & \\
x_1 & \dots & x_M & \dots & x_C & & \\
& & & & \vdots & & \\
& & & & y_R & &
\end{array}$$

unde șirurile $\{X_C\}$ și $\{Y_R\}$ conțin pixelii aflați pe orizontală și respectiv verticală din fereastră. Pseudomedianul poate fi definit ca:

$$pmed = \frac{1}{2} \max[\maximin\{X_C\}, \maximin\{Y_R\}] + \frac{1}{2} \min[\minimax\{X_C\}, \minimax\{Y_R\}].$$

Aplicații

1. Să se implementeze filtrele de zgomot trece-jos, *outlier*, median și pseudomedian;
2. Să se implementeze filtrul de zgomot care înlocuiește intensitatea fiecărui pixel din imagine cu elementul central al secvenței care conține intensitățile sortate ale pixelilor vecini.

6. Accentuarea contururilor

6.1. Filtrul Trece-Sus (FTS)

Accentuarea contururilor se poate realiza prin utilizarea filtrelor trece-sus. Cele mai utilizate măști FTS de mărime 3x3 sunt următoarele:

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad H_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad H_3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}.$$

Așa cum se poate observa, măștile utilizate pentru accentuarea contururilor au proprietatea că suma elementelor este 1, evitând în felul acesta modificări ale intensităților în imaginea procesată. Algoritmul constă în înlocuirea intensităților cu sumele obținute prin convoluția imaginii cu una din măștile FTS. Evident, este necesară și o normalizare a noilor intensități la intervalul [0, 255].

6.2. Tehnica ‘unsharp masking’

Noua intensitate a unui pixel reprezintă diferența ponderată dintre intensitatea originală (din imaginea cu rezoluție normală) și intensitatea pixelului corespunzător din imaginea cu rezoluție scăzută. Imaginea cu rezoluție scăzută poate fi obținută prin filtrare trece-jos (vezi paragraful 5.1).

$$G(j, k) = \frac{c}{2c-1} \cdot F(j, k) - \frac{1-c}{2c-1} \cdot F_L(j, k)$$

unde

- $c \in \left[\frac{3}{5}, \frac{5}{6} \right]$, deci $c \in [0.6, 0.8]$;
- $F(j, k)$ – intensitatea originală (intensitatea pixelului din imaginea cu rezoluție normală);
- $F_L(j, k)$ – intensitatea obținută cu un FTJ (intensitatea pixelului din imaginea cu rezoluție scăzută);
- L – dimensiunea măștii FTJ.

Aplicații

Să se implementeze cele două tehnici de accentuare a contururilor prezentate în lucrare: filtrul trece-sus și respectiv tehnica ‘unsharp masking’. Algoritmul ‘unsharp masking’ se va implementa printr-o singură parcurgere a imaginii procesate!

Observație

La convoluția imaginii pentru accentuarea contururilor, se va lucra cu intensitățile originale și nu cu cele modificate.

7. Detecția conturilor

Modificările sau discontinuitățile de intensitate (amplitudine) dintr-o imagine constituie caracteristici fundamentale care pot indica prezența unor obiecte într-o imagine. Aceste discontinuități sunt denumite contururi. La toți operatorii prezentați în lucrare, intensitățile se înlocuiesc cu sumele obținute prin convoluția imaginii procesate cu măștile corespunzătoare.

7.1. Operatorul Kirsch

$$1) H_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad H_2 = [1 \ -1], \quad H_3 = [1 \ 0 \ -1];$$

unde H_1 este utilizat pentru detecția conturilor orizontale, în timp ce măștile H_2 și H_3 se folosesc pentru conturile verticale din imagine.

$$2) H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad H_3 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \quad H_4 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

$$G = \max\{suma(H_1), suma(H_2), suma(H_3), suma(H_4)\}$$

7.2. Operatorul Laplace

$$H_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad H_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix},$$

unde H_2 este operatorul Laplace propus de Prewitt.

7.3. Operatorul Roberts

$$P = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix},$$

$$R = k \cdot \sqrt{suma(P)^2 + suma(Q)^2} \quad (\text{ex. } k = 7).$$

7.4. Operatorul Prewitt

$$P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix},$$

$$R = \sqrt{suma(P)^2 + suma(Q)^2}.$$

7.5. Operatorul Sobel

$$P = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix},$$

$$R = \sqrt{\text{suma}(P)^2 + \text{suma}(Q)^2}.$$

7.6. Operatorul Frei-Chen

$$F_1 = \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}, \quad F_2 = \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}, \quad F_3 = \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}, \quad F_4 = \begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}$$

$$F_5 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad F_6 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}, \quad F_7 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}, \quad F_8 = \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}, \quad F_9 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$R = \sqrt{\frac{\sum_{i=1}^4 \text{suma}(F_i)^2}{\sum_{i=1}^9 \text{suma}(F_i)^2}} \cdot 255.$$

Aplicații

Să se implementeze tehnicile de detecție a conturilor prezentate în lucrare. Notația $\text{suma}(M)$ reprezintă suma obținută prin convoluția imaginii procesate cu masca M .

8. Detecția de contur bazată pe filtrul Gabor

Diferența importantă față de metodele de detecție aconturului prezentate în lucrarea precedentă, este că filtrul Gabor determină dinamic matricea pentru fiecare fereastră de 3x3 din imagine. Cu alte cuvinte, se lucrează cu o matrice care se adaptează dinamic la context, și nu cu una prestabilită. Filtrul lui Gabor constă în modulația unei funcții gaussiene și a unui semnal sinusoidal [1]. În algoritm, $suma(P)$ și $suma(Q)$ reprezintă sumele obținute prin convoluția imaginii cu măștile P respectiv Q :

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad Q = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Algoritmul:

pentru fiecare fereastră de 3x3 din imagine (unde (row, col) este pixelul din mijlocul ferestrei)

- se calculează cele două sume: $suma(P)$ și $suma(Q)$;

dacă $suma(Q) = 0$

dacă $suma(P) \geq 0$, atunci $u = \frac{\pi}{2}$;

dacă $suma(P) < 0$, atunci $u = -\frac{\pi}{2}$;

altfel

$u = \arctg\left(\frac{suma(P)}{suma(Q)}\right)$;

dacă $suma(Q) < 0$, atunci $u = u + \pi$;

- $u = u + \frac{\pi}{2}$;

- $suma = 0$;

pentru fiecare pixel (r, c) din fereastră (r-linia, c-coloana)

• $scale = \left(e^{-\frac{poz(r)^2 + poz(c)^2}{2\sigma^2}} \right) \cdot \sin(\omega \cdot (poz(r) \cdot \cos u + poz(c) \cdot \sin u))$

//scale – elementele noii matrici (3x3).

• $suma = suma + scale \cdot Intensitate(r, c)$;

//poz(r) – poziția pe verticală față de primul element al ferestrei, $poz(r) \in [0, 2]$

//poz(c) – poziția pe orizontală față de primul element al ferestrei, $poz(c) \in [0, 2]$

- se setează pe poziția (row, col) intensitatea $suma$.

Observație: $\pi = 3.14$, $\sigma = 0.66$, $\omega = 1.5$.

Aplicație: să se implementeze filtrul Gabor.

9. Segmentarea imaginilor

Segmentarea este unul din cei mai importanți pași în analiza unei imagini [1]. Scopul este acela de a împărți imaginea în regiuni ce au o corelație puternică cu obiectele sau suprafețele din imagine. Există două tipuri de segmentare [1]: completă și parțială. Segmentarea completă generează un set de regiuni disjuncte ce corespund în mod unic cu obiectele din imagine. Pentru realizarea unei segmentări complete, este necesară cooperarea cu nivelele superioare de procesare, care utilizează cunoștințe specifice din domeniu. În cazul segmentării parțiale, imaginea este împărțită în regiuni disjuncte care sunt omogene relative la o anumită proprietate cum ar fi luminozitatea, culoarea, contextul reflectivitatea, etc.

Ambiguitatea prezentă în imagini este principala problemă a segmentării și ea este deseori însoțită de zgomot. Tehnicile de segmentare pot fi clasificate în patru categorii:

- filtrare locală și fixare de praguri (*threshold*);
- metode bazate pe *snake* și de tip balon, algoritmul *watershed*;
- tehnici de *region growing* și *split and merge*;
- metode de optimizare globală bazate pe funcții de energie, funcții bayesiene sau pe criteriul MDL (Minimum Description Length).

Metoda filtrării folosește informația locală și nu poate garanta contururi continue. Metodele de tip balon sau cele bazate pe *snake* folosesc doar informația existentă de-a lungul conturului și necesită o bună estimare a poziției de inițializare pentru obținerea unei convergențe corecte. Avantajul metodelor de *region growing* este acela că se bazează pe statistici realizate în interiorul fiecărei regiuni. Cele trei tehnici nu folosesc informația existentă la nivelul întregii imagini. Spre deosebire de acestea, metodele bazate pe energie, bayes sau MDL, utilizează criterii globale, dar fiind bazate pe o minimizare, minimumul este de multe ori greu de atins.

Tehnica *region growing* detectează o regiune identificând acei pixeli conectați la punctul de pornire care au intensități “similare”. Algoritmul recursiv asigură extinderea regiunii în patru direcții (spre cei patru pixeli vecini), dacă intensitățile acestora se încadrează în intervalul $[m-t, m+t]$, unde m este intensitatea medie a regiunii, iar t este o valoare prag introdusă de utilizator (parametru). Un alt parametru care trebuie introdus de utilizator este punctul de pornire care poate fi introdus printr-un click al mouse-ului pe imagine sau prin precizarea directă a coordonatelor acestuia. Ieșirea programului constă într-o copie binară a imaginii procesate, în care, un 1 reprezintă un pixel al obiectului detectat, iar un 0 indică un pixel care aparține fundalului.

Un alt algoritm *region growing* împarte imaginea în blocuri de aceeași dimensiune. Se recomandă divizarea în blocuri de 2×2 în cazul în care se aplică direct algoritmul *region growing*, respectiv de 16×16 dacă se efectuează și etapa *merge-split*.

În etapa opțională *merge-split* se folosește un prag (threshold) introdus de utilizator. Valoarea acestui prag determină care blocuri pot fi combinate (unite) într-un singur bloc și care blocuri pot fi divizate în blocuri mai mici, în funcție de diferența dintre intensitatea maximă și minimă din fiecare bloc. Dacă diferența *max-min* a unui bloc este apropiată de diferența *max-min* a unui bloc vecin (diferența *max-min* aferentă celor două blocuri este sub valoarea pragului), atunci blocurile se unesc într-unul singur. Un bloc este divizat ($1/2$ sau $1/4$) dacă diferența *min-max* este peste prag. Procesul *merge-split* se aplică recursive până când nici un bloc nu satisface criteriul de a fi divizat sau unit. Astfel, un bloc a cărui diferență *max-min* depășește valoarea pragului, va fi divizat până când diferențele *max-min* aferente blocurilor obținute sunt sub prag,

sau dimensiunile acestora ajung la un pixel, în acest caz diferența *max-min* fiind zero. Pentru evitarea divizării imaginii într-un număr prea mare de regiuni foarte mici, se poate folosi un parametru (introdus de utilizator) care precizează dimensiunea minimă a blocurilor obținute prin procesul de divizare. Astfel, utilizatorul poate forța terminarea algoritmului de segmentare cu un număr mic de regiuni, deoarece blocurile generate nu vor fi mai mici decât dimensiunea specificată.

Etapa *region growing* analizează fiecare bloc unindu-l cu blocuri adiacente care satisfac un anumit criteriu. Un criteriu constă în diferența *max-min* și determină aderarea acelor blocuri adiacente la o regiune, ale căror diferență *max-min* se încadrează cu o anumită toleranță (specificată de utilizator) în diferența *max-min* a regiunii. Această toleranță nu trebuie să fie identică cu pragul folosit în etapa *merge-split*. Un alt criteriu care determină care blocuri pot fi unite, constă în intensitatea medie a blocurilor. În acest caz, algoritmul asigură aderarea unui bloc la o regiune, dacă intensitatea medie a blocului se încadrează în intervalul $[m-t, m+t]$, unde m este intensitatea medie a regiunii, iar t este o valoare prag introdusă de utilizator (parametru).

Etapa *dissolve* combină regiunile care au o dimensiune mai mică decât un prag (stabilit de utilizator) cu regiunea adiacentă având cea mai apropiată medie a intensităților. În felul acesta se reduce numărul de regiuni, cele mai puțin semnificative fiind eliminate.

Aplicații

Să se implementeze, la alegere, unul din cei doi algoritmi de segmentare *region growing*, prezentați în lucrare.

10. Corelația imaginilor

Coeficientul de corelație este o metrică care exprimă similaritatea (nivelul de potrivire) dintre două semnale, de aceea, este foarte des folosit pentru căutarea șabloanelor (template matching). Coeficientul de corelație r al perechii (x, y) se calculează folosind următoarea formulă:

$$r_{xy} = \frac{\sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \frac{\sum_{i=1}^n y_i}{n}}{\sqrt{\left(\sum_{i=1}^n x_i^2 - \frac{\left(\sum_{i=1}^n x_i \right)^2}{n} \right) \left(\sum_{i=1}^n y_i^2 - \frac{\left(\sum_{i=1}^n y_i \right)^2}{n} \right)}}$$

Următorul pseudo-cod prezintă algoritmul care determină coeficientul maxim de corelație (cea mai bună potrivire) aferent unei perechi de imagine/șablon:

```
template_size = M * M;
sum = 0;
sqrsum = 0;
for j = 0 to M-1
{
    for i = 0 to M-1
    {
        sum += template[j, i];
        sqrsum += template[j, i] * template[j, i];
    }
}
mean_template = sum / template_size;
var2template = sqrsum - (sum * sum) / template_size;

for y = 0 to N-M //image rows
{
    for x = 0 to N-M //image columns
    {
        sum = 0;
        sqrsum = 0;
        prodsum = 0;
        for j = 0 to M-1 //template rows
        {
            for i = 0 to M-1 //template columns
            {
                sum += image[y+j, x+i];
                sqrsum += image[y+j, x+i] * image[y+j, x+i];
                prodsum += image[y+j, x+i] * template[j, i];
            }
        }
        var2image = sqrsum - (sum * sum) / template_size;
        corr_coeff[y,x] = (prodsum - (mean_template * sum)) / sqrt(var2image * var2template);
    }
}
}
```

unde *image* și *template* sunt imagini bidimensionale de mărime $N*N$ respectiv $M*M$, iar *var2image* este suma pătratelor distanțelor dintre intensitățile pixelilor din imagine acoperite de șablon. Așa cum se poate observa, valorile *mean_template*, *var2template* respectiv *template_size* pot fi precalculate și folosite pentru toți coeficienții de corelație dintr-o pereche imagine/șablon. Algoritmul poate fi folosit și pentru estimarea locală a mișcării (vezi următoarea lucrare de laborator).

Aplicații

Să se implementeze algoritmul prezentat în lucrare. Pentru testare se va alege un set de imagini și o imagine șablon. Pentru fiecare imagine se va determina coeficientul maxim de corelație cu imaginea șablon.

11. Algoritmi *block matching*

Algoritmii *block matching* [1, 2] sunt cele mai cunoscute metode de măsurare a similarității dintre două imagini și sunt folosite pentru estimarea mișcării. Aceste metode au fost adoptate de diverse standarde de codificare a imaginilor video. Metodele *block matching* presupun faptul că un bloc de pixeli are aceeași mișcare de translație de la un cadru la altul. Cele mai utilizate metode de comparare a blocurilor (măsurile de similaritate) au la bază suma pătratelor diferențelor SSD (Sum of Squared Differences), respectiv, suma diferențelor absolute SAD (Sum of Absolute Differences). De multe ori se preferă utilizarea metodei SAD datorită nivelului mic de complexitate (mai ales în implementările hardware).

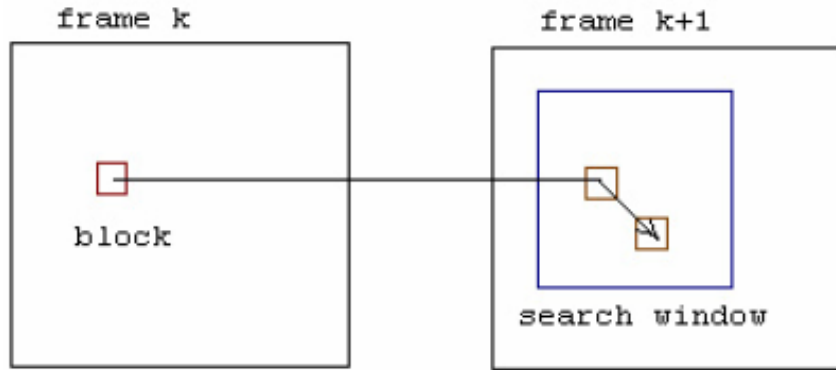


Figura 13. Algoritmul *block matching*

Considerând blocuri de dimensiune $N \times N$, și un vector de deplasament (u, v) al unui bloc candidat relativ la blocul șablon, cele două metrice de similaritate pot fi exprimate în felul următor:

$$SAD_{(u,v)} = \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} |f_t(i, j) - f_{t-1}(i+u, j+v)|$$

$$SSD_{(u,v)} = \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} (f_t(i, j) - f_{t-1}(i+u, j+v))^2$$

Algoritmul *Full-Search* (FS) compară un anumit bloc B cu toate blocurile din fereastra de căutare și găsește vectorul de mișcare optim dintre toți vectorii de mișcare din interiorul ferestrei. De aceea, acest algoritm necesită un număr mare de calcule. Din cauza complexității ridicate, algoritmul nu poate fi folosit în aplicații de timp real.

Algoritmul de căutare în trei pași (*Three Step Search*), propusă de Koga în 1981, se bazează pe o abordare a căutării de la grosier la fin, ceea ce duce la o scădere logaritmică la fiecare pas [1]. Așa cum se poate observa în Figura 14, la fiecare pas se verifică nouă puncte, iar punctul de minim al unei măsurii devine centrul de plecare al pasului următor. Căutarea se începe cu punctele marcate cu 0 (centrul de căutare) iar apoi cu cele marcate cu 1. Dacă în punctul 0 se obține similaritate maximă, atunci blocul este considerat static. Dacă într-unul din punctele marcate cu 1 se obține similaritate maximă, atunci acel punct devine noul centru, iar căutarea va continua în punctele marcate cu 2, și așa mai departe. Distanța dintre punctele evaluate se înjumătățește la fiecare pas al algoritmului.

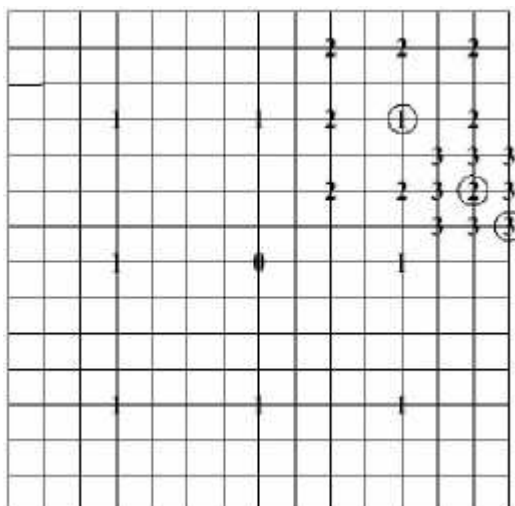


Figura 14. Algoritm de căutare în trei pași

Alți algoritmi care reduc gradul de complexitate sunt: căutarea în patru pași, căutarea logaritmică 2-D, căutarea ortogonală, căutarea în cruce, căutarea *gradient descent*.

Aplicații

Să se implementeze și să se compare cei doi algoritmi de block matching prezentați în lucrare: *Full Search* respectiv *căutarea în trei pași*.

Bibliografie

- [1] Brad R., *Procesarea Imaginilor și Elemente de Computer Vision*, Editura Universității “Lucian Blaga” din Sibiu, 2003.
- [2] Ho H., *2D-3D Block Matching*, MSc Thesis, University of Auckland.
- [3] <http://www.cee.hw.ac.uk/hipr/>
- [4] <http://www.ittc.ku.edu/~jgauch/research/kuim/html/00.00.html>.