Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Tehnici Avansate de Prelucrarea şi Analiza Imaginilor

Curs 5 – Filtrarea liniară

Ş.I. Bogdan IONESCU
Prof. Constantin VERTAN
Conf. Mihai CIUC

Plan Curs 5 – Filtrarea liniară

5.1. Introducere operaţii de vecinătate şi filtrare

5.2. Filtrarea liniară de netezire

5.3. Filtrarea liniară de derivare

5.1. Introducere operaţii de vecinătate

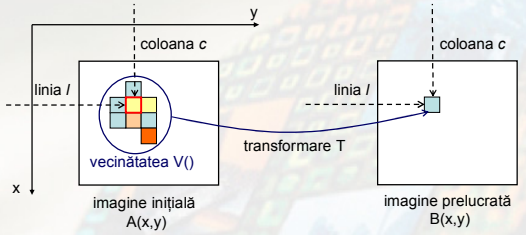
Operaţii de vecinătate

- fiecare pixel din imaginea modificată depinde de un anumit număr (ex. o fereastră) de pixeli din imaginea iniţială, situaţi în vecinătatea acestuia.

→*în cazul operaţiilor punctuale*, nu se lua în calcul "contextul" valorilor pixelilor, astfel că doi pixeli de valori identice erau consideraţi identici,
(vezi limitări operaţii histogramă)

→ *în cazul operaţiilor de vecinătate*, pixelii depind de vecini, astfel chiar dacă aceştia au valori identice, pot aparţine unor zone diferite din imagine,
(ex. discontinuitate vs. regiune uniformă)

Definiţie operaţii de vecinătate



$$B(l,c) = T(A(V_{(l,c)}))$$

unde *A()* = imaginea iniţială, *B()* = imaginea modificată şi $V_{(l,c)}$ reprezintă o vecinătate a pixelului de coordonate *(l,c)*

Definiţie operaţii de vecinătate

$$B(l,c) = T(A(V_{(l,c)}))$$

unde *A()* = imaginea iniţială, *B()* = imaginea modificată şi $V_{(l,c)}$ reprezintă o vecinătate a pixelului de coordonate *(l,c)*

transformarea este specificată integral dacă:
- se specifică vecinătatea $V_{(l,c)}$ (fereastră de prelucrare)
- se specifică modul de combinare al pixelilor vecini, funcţia *T()*.

→ în funcţie de *T()*, operaţiile de filtrare (de vecinătate) sunt:

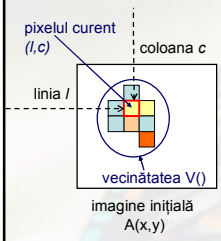
↳ **filtrări liniare**: *T()* este o combinaţie liniară a valorilor pixelilor din vecinătate.

↳ **filtrări neliniare**: *T()* este o funcţie neliniară:

- intrinsec neliniară (filtrare neliniară),

- neliniară ca efect al adaptării la conţinut (filtrare adaptivă),

Specificarea vecinătății



vecinătate = o mulțime de pixeli din planul imaginii, vecini cu pixelul curent (prelucrat)

definirea vecinătății = specificarea pozițiilor în imagine ale pixelilor consideraţi vecini, relativ la coordonatele *(l,c)* (pixel curent)

$$V_{(l,c)} = \{(m_1,n_1),(m_2,n_2),...,(m_K,n_K)\}$$

unde (m_i,n_i), cu i=1,...,K, reprezintă coordonatele relative la *(l,c)* iar *K* reprezintă numărul de pixeli din vecinătate.

Specificarea vecinătății

coordonate relative:

$$V_{(l,c)} = \{(m_1,n_1),(m_2,n_2),...,(m_K,n_K)\}$$

unde (m_i,n_i), cu i=1,...,K, reprezintă coordonatele relative la *(l,c)* iar *K* reprezintă numărul de pixeli din vecinătate.

coordonate imagine:

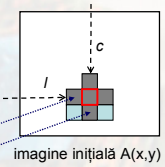
$$V_{(l,c)} = \{(l+m_1,c+n_1),...,(l+m_K,c+n_K)\}$$

unde *$(l+m_i,c+n_i)$*, cu *i=1,...,K*, reprezintă coordonatele din sistemul de coordonate ataşat imaginii.

Tipuri de vecinătăți

vecinătatea V_4

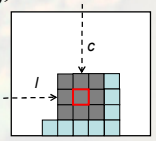
= vecinii Nord, Est, Sud, Vest (în stea)
→ pixelii colorați cu gri sunt vecinii pixelului curent *(l,c)*



$$V_4 = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}, K = 5$$

vecinătatea V_8

= toate punctele cardinale (cei opt vecini ce formează un cerc discret)

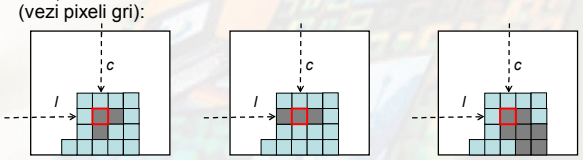


$$V_8 = \begin{Bmatrix} (0,0),(0,1),(0,-1),(1,0),(-1,0), \\ (1,-1),(-1,1),(-1,-1),(1,1) \end{Bmatrix} \quad K = 9$$

Tipuri de vecinătăți

→ în definirea vecinătății se specifică şi pixelul curent (anumite operații de filtrare pot calcula noua valoare a pixelului funcție sau nu de însăşi valoarea pixelului curent)

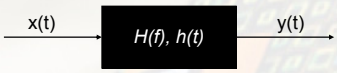
→ tipurile de vecinătăți nu se limitează doar la V_4 sau V_8 , în funcție de necesitate utilizatorul poate defini orice vecinătate (vezi pixeli gri):



$$B(l,c) = T(A(V_{(l,c)}))$$
$$B(l,c) = T(A(l+m_1,c+n_1),...,A(l+m_K,c+n_K))$$

Operația de filtrare în general

prelucrare de semnal:



unde *x(t)* este semnalul de intrare iar *y(t)* este semnalul filtrat,

- filtrul (cutia neagră) este caracterizat de funcțiile *H(f)* (funcția de transfer în frecvență) şi/sau *h(t)* (răspunsul la impulsul unitate)

$$H(f) = Fourier(h(t))$$
$$Y(f) = H(f) \cdot X(f)$$

→ filtrarea înseamnă:
în temporal convoluție,
în frecvență produs,

$$y(t) = (h * x)_{(t)} = \int_{-\infty}^{\infty} h(t-u)x(u)du = \int_{-\infty}^{\infty} h(u)x(t-u)du$$

Operația de filtrare a imaginilor

prelucrare de imagini: → avem de-a face cu semnale bidimensionale discrete

$$B(l,c) = T(A(l+m_1,c+n_1),...,A(l+m_K,c+n_K))$$

unde A() = imaginea inițială, B() = imaginea modificată.

- dacă vorbim de filtrare liniară atunci se poate scrie ca sumă ponderată:

$$B(l,c) = \sum_{(m,n)\in V_{(l,c)}} w_{m,n} \cdot A(l+m,c+n)$$

unde $V_{(l,c)}$ reprezintă vecinătatea considerată iar $w_{m,n}$ reprezintă ponderile filtrului (valori reale, $w_{m,n}$ = pondere A(l+m,c+n)),

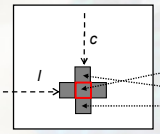
identificăm filtrarea clasică ⇨ x(t) ~ A(x,y) , y(t) ~ B(x,y), *h(t) ~ $w_{m,n}$,* *y(t)=$(h*x)_{(t)}$ ~ B(l,c)=$(w*A)_{(l,c)}$*

Operația de filtrare a imaginilor

→ deci, filtrarea liniară a unei imagini înseamnă un produs de convoluție cu ponderile filtrului considerat (răspunsul impulsional discret).

filtrul este specificat integral dacă:
- se specifică vecinătatea $V_{(l,c)}$ (fereastră de prelucrare)
- se specifică ponderile filtrului, $w_{m,n}$

→ considerăm exemplul următor: $K = 5$



imagine inițială A(x,y)

$$V_4 = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}$$

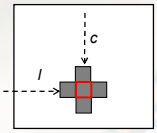
$w_{0,0} = 1/2$ → pondere pixel *A(l+0,c+0)*

$w_{0,1} = w_{0,-1} = w_{1,0} = w_{-1,0} = 1/8$

→ ponderi pixeli *A(l+0,c+1), A(l+0,c-1),...*

Operația de filtrare a imaginilor

→ exemplu (continuare): $V_4 = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}$



imagine inițială A(x,y)

→ prin analogie cu poziția pixelilor în imagine, coeficienții filtrului pot fi reprezentați ca o matrice:

$$\begin{bmatrix} 0 & 0.125 & 0 \\ 0.125 & 0.5 & 0.125 \\ 0 & 0.125 & 0 \end{bmatrix}$$

($w_{-1,0}$ deasupra, $w_{0,-1}$ $w_{0,0}$ $w_{0,1}$ pe linia centrală, $w_{1,0}$ dedesubt)

marcăm originea

completăm matricea cu valori nule

→ forma matriceală se numește *mască de filtrare*, iar filtrarea (convoluția) unei imagini se rezumă la aplicarea acestei măști imaginii.

$$B(l,c) = \sum_{(m,n)\in V} w_{m,n} \cdot A(m+l, n+c)$$

Operația de filtrare a imaginilor

→ exemplu (continuare):

$$V_4 = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}$$

$$\begin{bmatrix} 0 & 0.125 & 0 \\ 0.125 & 0.5 & 0.125 \\ 0 & 0.125 & 0 \end{bmatrix}$$

($w_{-1,0}$ deasupra, $w_{0,-1}$ $w_{0,0}$ $w_{0,1}$ pe linia centrală, $w_{1,0}$ dedesubt)

masca de filtrare

$$B(l,c) = \sum_{(m,n)\in V} w_{m,n} \cdot A(m+l, n+c)$$

$$B(l,c) = w_{0,0} \cdot A(l+0, c+0) +$$
$$w_{1,0} \cdot A(l+1, c+0) + w_{0,1} \cdot A(l+0, c+1) +$$
$$+ w_{-1,0} \cdot A(l-1, c+0) + w_{0,-1} \cdot A(l+0, c-1)$$

$$B(l,c) = \frac{1}{2} \cdot A(l,c) + \frac{1}{8} \cdot [A(l,c+1) + A(l,c-1) + A(l-1,c) + A(l+1,c)]$$

Operația de filtrare a imaginilor

$$B(l,c) = \sum_{(m,n)\in V} w_{m,n} \cdot A(m+l, n+c)$$

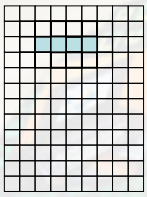
→ ținând cont de cele menționate anterior, filtrarea liniară a unei imagini se realizează simplificat pe baza următorului algoritm:

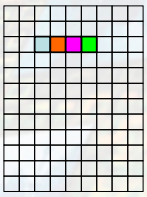
1. imaginea *A()* este parcursă pixel cu pixel, de la stânga la dreapta şi de sus în jos,
2. pentru pixelul curent *A(l,c)* se poziționează masca de filtrare cu centrul în acest punct,
3. coeficienții măştii sunt înmulțiți cu valorile pixelilor corespunzători din imaginea *A()*,
4. suma valorilor obținute va reprezenta valoarea pixelului *B(l,c)* din imaginea filtrată.

atenție: inițial imaginea *B()* nu conține nici o valoare !

Operația de filtrare a imaginilor

→ reprezentare grafică algoritm:

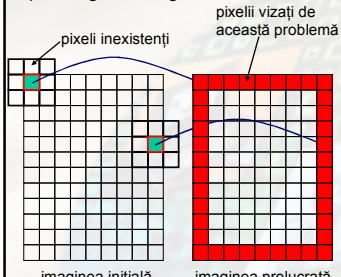

imagine inițială A(x,y)


imagine filtrată B(x,y)

→ se mai numeşte principiul "ferestrei glisante".

Probleme de implementare practică

→ imaginea nu este infinită, este mărginită; cum filtrăm pixelii de pe marginile imaginii ?



imaginea inițială

imaginea prelucrată

soluții:

- ignorarea liniilor şi a coloanelor cu problemă (rezoluție ↗ OK)
- bordarea virtuală a imaginii cu suficiente linii şi coloane pentru a furniza valorile lipsă,

valori ? (ex. replicare pixeli, prelucrare circulară, pixeli constanți, etc.)

Probleme de implementare practică

→ complexitatea de calcul = numărul de operații elementare efectuate (ex. adunări, înmulțiri, etc.)

→ imaginea = cantitate mare de date → trebuie minimizată complexitatea de calcul.

- rezoluție de pixeli: M x N
- filtrare cu V_X: $B(l,c) = \sum_{(m,n)\in V_X} w_{m,n} \cdot A(m+l, n+c)$

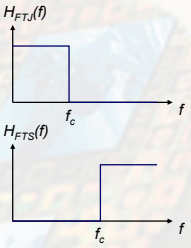
→ K înmulțiri + K-1 adunări = 2K-1

- complexitate de ordinul *O(M*N*(2*K-1))*

→ complexitatea de calcul depinde în principal de dimensiunea imaginii şi apoi de dimensiunea ferestrei de filtrare (vecinătate)

→ pentru ferestre mari este preferabilă aplicarea iterativă a unor filtre de dimensiuni mai mici echivalente (descompunere)

Tipuri de filtrări liniare



prelucrare de semnal:

- FTJ : filtru trece jos → sunt înlăturate frecvențele înalte

- FTS : filtru trece sus → sunt înlăturate frecvențele joase şi componenta continuă

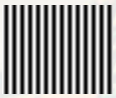
prelucrare de imagini: → aceleaşi concepte dar au o interpretare diferită datorită informației spațiale (de vecinătate):

ce înseamnă frecvență joasă?

⇨ frecvența ↘ proporțional cu ↗ uniformității imaginii

Tipuri de filtrări liniare

ce înseamnă frecvență înaltă ? ~ contururi



⇨ frecvența ↗ proporțional cu ↘ uniformității imaginii

→ imaginea poate fi descompusă în imagini de frecvență (transformata Fourier, Cosinus, etc.)

→ operațiile de filtrare sunt:

- creşterea uniformității în interiorul regiunilor sau **filtrare de netezire** (~ FTJ)

- creşterea contrastului pe frontierele regiunilor ce are la bază o **filtrare de derivare** (~ FTS)

5.2. Filtrarea de netezire

Filtrarea de netezire (FTJ)

- creşterea uniformității în interiorul regiunilor ~ reducerea micilor variații ale valorilor pixelilor, datorate, de exemplu, zgomotului.

- unul dintre tipurile de zgomot cel mai des întîlnite îl reprezintă *zgomotul alb Gaussian aditiv* (ZAGA):

$$A(l,c) = A_0(l,c) + z(l,c)$$

unde *A_0() este imaginea neafectată de zgomot iar z()* reprezintă zgomotul alb

$$z(l,c) \leftarrow N(0,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{1}{2\sigma^2}(x^2+y^2)}$$

= distribuție Gaussiană (normală) de medie 0, necorelat, densitate spectrală de putere constantă

Filtrarea de netezire (FTJ)

- exemple de imagini afectate de zgomot alb:

creşte σ^2

- filtrul optim de netezire în acest caz este **filtrul de mediere**:

$$\left.\begin{aligned} B(l,c) &= \sum_{(m,n)\in V} w_{m,n} \cdot A(l+m, c+n) \\ Card(V) &= K \end{aligned}\right\} \; w_{m,n} = \frac{1}{K}$$

Filtrarea de netezire (FTJ)

- exemple de filtre de mediere:

V_4: $K = 5 \rightarrow w_{m,n} = \frac{1}{5} \rightarrow \begin{bmatrix} 0 & 1/5 & 0 \\ 1/5 & 1/5 & 1/5 \\ 0 & 1/5 & 0 \end{bmatrix}$ masca de filtrare

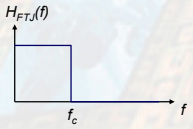
V_8: $K = 9 \rightarrow w_{m,n} = \frac{1}{9} \rightarrow \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$ masca de filtrare

- etc., se pot determina coeficienții pentru orice vecinătate

Filtrarea de netezire (FTJ)

- medierea de ce este filtru FTJ ?

componenta continuă trece prin filtru nealterată



$$B(l,c) = \sum_{(m,n)\in V_X} w_{m,n} \cdot A(l+m, c+n)$$

$$A(l+m, c+n) = ct, \forall (m,n) \in V_X, \forall (l,c)$$

$$B(l,c) = ct \cdot \sum_{(m,n)\in V_X} w_{m,n} = ct$$

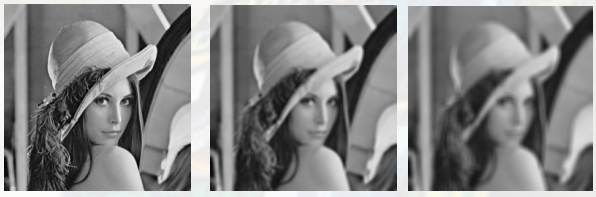
$$\sum_{(m,n)\in V_X} w_{m,n} = 1, \; w_{m,n} > 0$$

FTJ = suma coeficienților filtrului trebuie să fie 1

$\begin{bmatrix} 0 & 1/5 & 0 \\ 1/5 & 1/5 & 1/5 \\ 0 & 1/5 & 0 \end{bmatrix}$ $\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$

Filtrarea de netezire (FTJ)

- exemple:



imagine inițială | mediere fereastră 5x5 | mediere fereastră 9x9

- filtrarea de mediere are ca efect începțoşarea imagini (blur), care este cu atât mai puternică cu cât vecinătatea este mai mare.

Filtrarea de netezire (FTJ)

imagine inițială

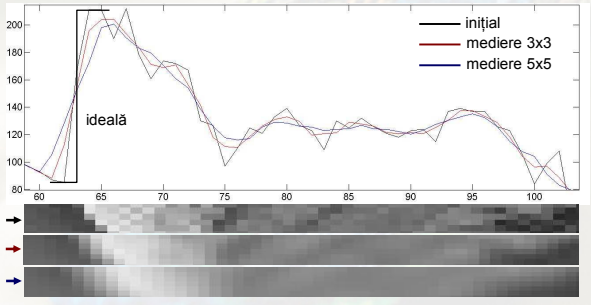
- exemple, acum şi cu zgomot:

imagine afectată de zgomot alb | mediere fereastră 3x3 | mediere fereastră 5x5

- zgomotul este diminuat proporțional cu creşterea dimensiunii vecinătății (puterea zgomotului este redusă de K ori) → compromis!

Filtrarea de netezire (FTJ)

- efect nedorit = blur şi anume începțoşarea frontierelor:



secțiunea transversală a liniei 100 din imagine (oX coordonate pixeli, oY nivel de gri)

Filtrarea de netezire (FTJ)

- filtre de **mediere ponderată**: vecinii au ponderi diferite dar suma este 1 (FTJ)

$$B(l,c) = \sum_{(m,n)\in V} w_{m,n} \cdot A(l+m, c+n)$$

masca 1: $\begin{bmatrix} 0 & 0.125 & 0 \\ 0.125 & 0.5 & 0.125 \\ 0 & 0.125 & 0 \end{bmatrix}$

masca 2: $\begin{bmatrix} 0.05 & 0.1 & 0.05 \\ 0.1 & 0.4 & 0.1 \\ 0.05 & 0.1 & 0.05 \end{bmatrix}$

masca 3: $\begin{bmatrix} 0 & 0 & 0.15 \\ 0 & 0.7 & 0 \\ 0.15 & 0 & 0 \end{bmatrix}$

masca 4: $\begin{bmatrix} 0.15 & 0 & 0.15 \\ 0 & 0.4 & 0 \\ 0.15 & 0 & 0.15 \end{bmatrix}$

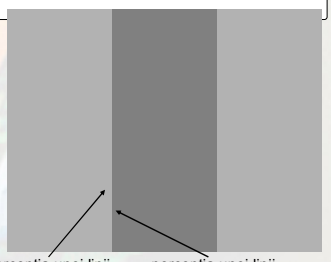
5.3. Filtrarea de derivare

Filtrarea de derivare (FTS)

>contrastarea unei imagini (vezi operaţii punctuale) are ca obiectiv îmbunătăţirea percepţiei vizuale a contururilor obiectelor (îmbunătăţirea detectabilităţii componentelor scenei de-a lungul frontierelor acestora).

>SVU are tendinţa de a adânci profilul zonelor de tranziţie dintre regiunile uniforme ("benzile lui Mach"):

aceste linii nu există în realitate fiind create artificial de SVU ←



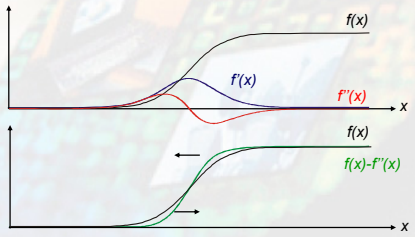
percepţia unei linii mai luminoase

percepţia unei linii mai întunecate

Filtrarea de derivare (FTS)

>acest fenomen are loc datorită prelucrărilor de tip derivativ ce apar în diferitele etape pe care le parcurge informaţia vizuală (vezi cursul1)

>efectul global poate fi descris prin scăderea din semnalul original a unei derivate secunde a acestuia.



Filtrarea de derivare (FTS)

> derivata de *ordinul I* (folosită de regulă pentru estimarea direcţiei gradientului contururilor → edge detection)

$$\begin{bmatrix} \textcircled{0} & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \textcircled{1} & 0 \\ 0 & \textcircled{-1} \end{bmatrix}$$

Roberts

$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & \textcircled{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 0 & \textcircled{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Prewitt

??

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & \textcircled{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & \textcircled{0} & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \textcircled{0} & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & \textcircled{0} & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$$

Sobel Isotropic

Filtrarea de derivare (FTS)

> derivata de *ordinul II* (folosită şi pentru detecţia contururilor)

→ se foloseşte operatorul Laplacian:

$$\nabla^2 A(x,y) = \frac{\partial^2 A(x,y)}{\partial x^2} + \frac{\partial^2 A(x,y)}{\partial y^2}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & \textcircled{4} & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

masca 1

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & \textcircled{8} & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

masca 2

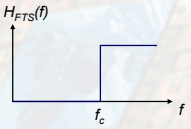
$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & \textcircled{4} & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

masca 3

Filtrarea de derivare (FTS)

- derivarea de ce este filtru FTS ?

componenta continuă este eliminată în urma filtrării



$$B(l,c) = \sum_{(m,n)\in V_X} w_{m,n} \cdot A(l+m, c+n)$$

$$A(l+m, c+n) = ct, \forall (m,n) \in V_X, \forall (l,c)$$

$$B(l,c) = ct \cdot \sum_{(m,n)\in V_X} w_{m,n} = 0$$

$$\sum_{(m,n)\in V_X} w_{m,n} = 0$$

FTS = suma coeficienţilor filtrului trebuie să fie 0

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

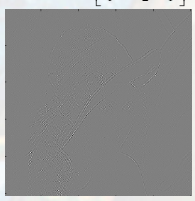
Filtrarea de derivare (FTS)

- exemple:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$


imagine inițială


Laplacian mască 1


Laplacian mască 3

-*observație*: în urma aplicării Laplacianului, valorile obținute pot fi negative sau mai mari ca 255 → rezultatul filtrării nu este o imagine în sensul clasic, ci o imagine a valorilor derivatei secunde,

(în exemple anterioare, strict pentru vizualizare, gama de valori obținute a fost rescalată în intervalul [0;255])

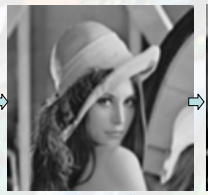
Filtrarea de derivare (FTS)

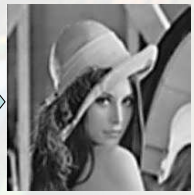
- creşterea contrastului pe baza derivate secunde

$$A(x,y) = A(x,y) - k \cdot \nabla^2 A(x,y)$$

unde *k* este coeficientul de multiplicare al Laplacianului (parametru de reglaj)


imagine inițială


imagine mediată
(mască 5x5)


creştere contrast
(*k=0.3*)

Sfârşit Curs