



LAPI – Laboratorul de
Analiza și Prelucrarea
Imaginilor



Universitatea
POLITEHNICA din
București



Facultatea de Electronică,
Telecomunicații și
Tehnologia Informației

Analiza și Prelucrarea Secvențelor de Imagini

Ș.I.dr.ing. Bogdan IONESCU

București, 2010

> M2. Filtrare spațio-temporală

- 2.1. [Introducere]
- 2.2. [Filtre liniare]
- 2.3. [Filtre statistice de ordine]
- 2.4. [Filtre multirezoluție]
- 2.5. [Filtrare artefacte]

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

1

> M2. Filtrare spațio-temporală [Introducere]

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

2

Îmbunătățirea calității video

obiectiv: atenuarea sau eliminarea diferitelor degradări, de regulă, ale informației vizuale;

↳ **zgomot:** perturbă informația utilă:






imagine inițială zgomot alb zgomot impulsiv zgomot multiplicativ

> în unele cazuri se pot folosi tehnicile de la imagini statice,
[vezi Curs 5-6 http://imag.pub.ro/~bionescu/index_files/Page328.htm]





> metode ce iau în calcul corelația temporală: **filtre liniare temporale**,
filtre statistice de ordine și **filtre multirezoluție**;

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

3

Îmbunătățirea calității video

↳ **artefacte:** ex. conturarea blocurilor DCT datorită unui factor de compresie ridicat:

imagine inițială imagine comprimată (blocuri DCT) imagine inițială imagine cu "mosquito artefacts"

[P.-T. Le Dinh, J. Patry, Algorith]

> metode de filtrare atât în domeniul spațial cât și frecvențial,

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

4

Îmbunătățirea calității video

↳ **blotches:** puncte/regiuni albe sau negre ce apar de regulă pe benzile de film deteriorate fizic (praf, degradare gelatină, ...):





imagine inițială imagine deteriorată segmentare blotch

[M.K. Güllü, O. Urhan, S. Ertürk, Blotch Detection and Removal for Archive Video Restoration]

↳ **vinegar:** pierderea parțială a informației de culoare, încețoșare neuniformă (distrugere chimică – miros de oțet):




imagine inițială imagine deteriorată

[A.C. Bovik, The Essential Guide to Video Processing, 2009
sursă imagine RTP-Radiotelevisão Portuguesa]

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

5

Îmbunătățirea calității video

↳ **intensity flicker**: variații artificiale ale intensității luminoase în timp:



[P. M. B. van Roosmalen, R. L. Lagendijk, J. Biemond, Correction of Intensity Flicker in Old Film Sequences]

↳ **moiré**: în principal datorat scanării diferite (film original vs. numerizare), filmarea unui ecran, probleme de eșantionare (strobing - TV digital):



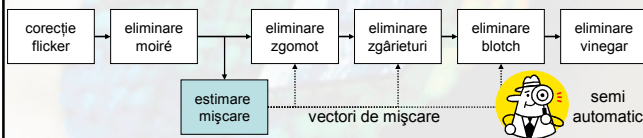
Îmbunătățirea calității video

↳ **zgârieturi**: linii luminoase sau întunecate datorate particulelor de praf ce zgârie filmul la proiecție (de regulă verticale și localizate):



[A.C. Bovik, The Essential Guide to Video Processing, 2009
sursă imagine
CNC - Archives Françaises du Film]

> exemplu linie de restaurare video [R.L. Lagendijk, J. Biemond, A. Rareș, M.J.T. Reinders, Video Enhancement and Restoration, 2009]:



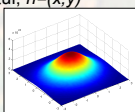
Filtrarea spațio-temporală a zgomotului

> de regulă, pentru a determina modul de filtrare, se va modela efectul agregat al diferitelor tipuri de zgomot ce afectează secvența:

$$g(n, k) = f(n, k) + w(n, k)$$

unde $g()$ este secvența de imagini înregistrată, $f()$ este secvența inițială considerată ideală (neperturbată), $w()$ reprezintă zgomotul, $n=(x, y)$ reprezintă coordonatele spațiale iar k indicele temporal,

$$w(n, k) \leftarrow N(0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2\sigma^2}(x^2+y^2)}$$



→ $w()$ zgomot alb de distribuție Gaussiană (normală) de medie 0, necorelat și densitate spectrală de putere constantă.

$$g(n, k) \xrightarrow{\text{filtru}} \hat{f}(n, k) \rightarrow f(n, k)$$

> M2. Filtrare spațio-temporală [Filtre liniare]

Filtrarea spațio-temporală a zgomotului

↳ **filtre liniare**

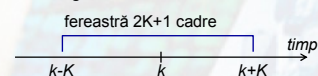
filtre de mediere temporală (netezire):

[R.L. Lagendijk, J. Biemond, A. Rareș, M.J.T. Reinders, Video Enhancement and Restoration, 2009]

A. **medierea ponderată** într-o fereastră temporală:

$$\hat{f}(n, k) = \sum_{l=-K}^K h(l) \cdot g(n, k-l)$$

unde $h()$ reprezintă coeficienții filtrului iar fereastra temporală conține $2K+1$ imagini consecutive.



$$\rightarrow \text{coeficienți egali } h(l) = \frac{1}{2K+1} = \text{mediere};$$

Filtrarea spațio-temporală a zgomotului

↳ **filtre liniare (continuare)**

filtre de mediere temporală (netezire):

B. **medierea ponderată optimă** într-o fereastră temporală:

$$h(l) \leftarrow \min E\{[f(n, k) - \hat{f}(n, k)]^2\}$$

= filtrul Wiener optimal.

$$\begin{bmatrix} h(-K) \\ \dots \\ h(0) \\ h(1) \\ \dots \\ h(K) \end{bmatrix} = \begin{bmatrix} R_{gg}(0) & \dots & R_{gg}(-K) & \dots & \dots & R_{gg}(-2K) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ R_{gg}(K) & \dots & R_{gg}(0) & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & R_{gg}(0) & \dots \\ R_{gg}(2K) & \dots & \dots & \dots & \dots & R_{gg}(0) \end{bmatrix}^{-1} \begin{bmatrix} R_{fg}(-K) \\ \dots \\ R_{fg}(0) \\ R_{fg}(1) \\ \dots \\ R_{fg}(K) \end{bmatrix}$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare (continuare)

filtre de mediere temporală (netezire):

$$\begin{bmatrix} R_{gg}(0) & \dots & R_{gg}(-K) & \dots & R_{gg}(-2K) \\ R_{gg}(K) & \dots & R_{gg}(0) & \dots & R_{gg}(K) \\ R_{gg}(2K) & \dots & R_{gg}(K) & \dots & R_{gg}(0) \end{bmatrix} \times \begin{bmatrix} R_{gg}(-K) \\ R_{gg}(0) \\ R_{gg}(K) \end{bmatrix}$$

B. medierea ponderată optimală într-o fereastră temporală (continuare):

$$R_{gg}(m) = E\{g(n, k) \cdot g(n, k - m)\}$$

= funcția de autocorelație temporală a lui $g()$ cu $g()$.

$$R_{fg}(m) = E\{f(n, k) \cdot g(n, k - m)\}$$

= funcția de intercorelație temporală a lui $f()$ cu $g()$.

> cu cât fereastra este mai mare cu atât filtrarea este mai puternică (puterea zgomotului \searrow) \Rightarrow încețoșarea obiectelor în mișcare;

> se mediază de fapt pixeli diferiți dar care se află în aceeași poziție spațială la momentul k ;

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare (continuare)

filtre de mediere temporală (netezire):

C. medierea ponderată, cu compensarea mișcării:

$$\hat{f}((x, y), k) = \sum_{l=-K}^K h(l) \cdot g(x - d_x(x, y; k, l), y - d_y(x, y; k, l), k - l)$$

$\sim x$ $\sim y$

unde $\{d_x(x, y; k, l); d_y(x, y; k, l)\}$ reprezintă vectorul de mișcare al pixelului de coordonate (x, y) între cadrele la momentul k și l .

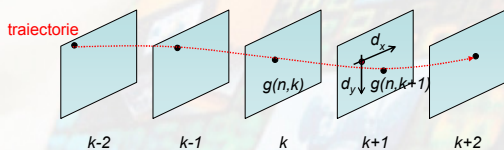
> *avantaj*: voi media același pixel pe măsură ce se deplasează în timp de la un cadru la altul;

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare (continuare)

filtre de mediere temporală (netezire):

C. medierea ponderată cu compensarea mișcării (continuare):



> cu cât crește dimensiunea ferestrei temporale, crește și potențialul reducerii zgomotului dar și probabilitatea apariției artefactelor datorate estimării incorecte a mișcării pentru distanțe temporale mari,

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare (continuare)

filtre de mediere temporală (netezire):

D. medierea ponderată cu aproximarea compensării mișcării:

> nu dispunem de informația de mișcare iar aceasta nu poate fi estimată deoarece secvența este foarte afectată de zgomot:

D.1. detectăm schimbările de la o imagine la alta și localizarea lor temporală:

$\rightarrow h()$ va fi adaptat spațial (ex. mediere minimă pt. schimbări \nearrow),

D.2. pentru fiecare coordonată spațială, filtrăm după M potențiale direcții de mișcare alese "a priori":

\rightarrow folosind un anumit criteriu alegem unul dintre rezultatele parțiale,

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare (continuare)

filtre de mediere temporală (netezire):

D.2. medierea ponderată cu aproximarea compensării mișcării (continuare):

$$\hat{f}_{i,j}((x, y), k) = \frac{1}{3} [g(x - i, y - j, k - 1) + g(x, y, k) + g(x + i, y + j, k + 1)]$$

unde i și j definesc direcțiile posibile iar filtrarea se face ca:

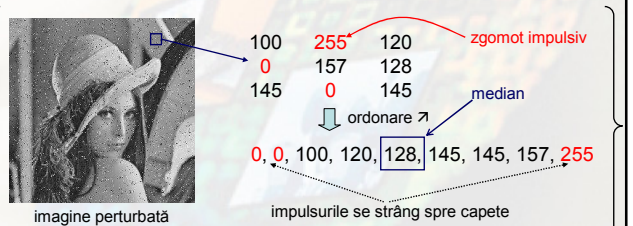
$$\hat{f}(n, k) = \text{median}\{\hat{f}_{-1,-1}(n, k), \hat{f}_{1,-1}(n, k), \hat{f}_{-1,1}(n, k), \hat{f}_{1,1}(n, k), \hat{f}_{0,0}(n, k), g(n, k)\}$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare (continuare)

filtre de mediere temporală (netezire):

D.2. medierea ponderată cu aproximarea compensării mișcării (continuare):



Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre de mediere temporală (netezire):

E. medierea ponderată spațio-temporal:

$$\hat{f}(n, k) = \sum_{(m, l) \in S} h(m, l) \cdot g(n - m, k - l)$$

unde S reprezintă un suport spațio-temporal (fereastră 3D), m sunt coordonatele spațiale iar l deplasarea temporală,



→ coeficienți egali sau pot fi optimizați → filtru Wiener 3D;

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre de mediere temporală (netezire):

E. medierea ponderată spațio-temporal (continuare):

$$\hat{f}(n, k) = \sum_{(m, l) \in S} h(m, l) \cdot g(n - m, k - l)$$

> limitări optimizare Wiener: nu dispunem de funcțiile de autocorelație iar ipoteza de staționaritate în sens larg nu este adevărată datorită mișcării obiectelor sau a efectelor vizuale;

→ coeficienți adaptivi:

$$h(m, l; n, k) = \frac{c}{1 + \max \{\alpha, [g(n, k) - g(n - m, k - l)]^2\}}$$

unde $h(m, l; n, k)$ ponderează intensitatea pixelului de coordonate $n-m$ la momentul temporal $k-l$ și contribuie la $\hat{f}(n, k)$,

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre de mediere temporală (netezire):

E. medierea ponderată spațio-temporal (continuare):

→ coeficienți adaptivi (continuare):

$$h(m, l; n, k) = \frac{c}{1 + \max \{\alpha, [g(n, k) - g(n - m, k - l)]^2\}}$$

→ dacă $[g(n, k) - g(n - m, k - l)]^2 < \alpha$, ~ valoarea pixelului se modifică foarte puțin (regiuni omogene temporal) ⇒ pondere importantă; $\frac{c}{1+} \rightarrow 0$

→ altfel, ~ valoare pixelului se schimbă semnificativ (mișcare importantă, diferite, ...), pondere mică = excludere; $\frac{c}{\rightarrow \infty}$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

[R.L. Lagendijk, J. Biemond, A. Rares, M.J.T. Reinders, Video Enhancement and Restoration, 2009]

> *avantaj*: necesită o fereastră de analiză mult mai mică, de doar câteva imagini (de regulă una singură),

> forma generală:

$$\hat{f}(n, k) = \hat{f}_b(n, k) + \alpha(n, k) \cdot [g(n, k) - \hat{f}_b(n, k)]$$

unde $\hat{f}_b(n, k)$ reprezintă predicția cadrului original la momentul k pe baza cadrelor filtrate anterior (~recurența), $\alpha(n, k)$ reprezintă termenul de actualizare a predicției raportată la cadrul $g()$, curent, $n=(x, y)$ sunt coordonatele spațiale iar k indicele temporal.

→ dacă $\alpha(n, k) = 1 \Rightarrow \hat{f}(n, k) = g(n, k)$ (fără filtrare),

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

$$\hat{f}(n, k) = \hat{f}_b(n, k) + \alpha(n, k) \cdot [g(n, k) - \hat{f}_b(n, k)]$$

A. alegerea lui $\hat{f}_b()$:

→ din cadrul anterior:

$$\hat{f}_{b1}(n, k) = \hat{f}(n, k - 1)$$

→ din cadrul anterior + compensarea mișcării:

$$\hat{f}_{b2}(n, k) = \hat{f}(n - d(n; k, k - 1), k - 1)$$

unde $d(n; k, k - 1)$ reprezintă vectorul de mișcare pentru pixelul de coordonate n între imaginile la $k-1$ și k .

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

$$\hat{f}(n, k) = \hat{f}_b(n, k) + \alpha(n, k) \cdot [g(n, k) - \hat{f}_b(n, k)]$$

B. alegerea lui $\alpha(n, k)$:

→ valoare globală, fixă (introduce artefacte în cazul mișcării = "comet-tails");

$$\rightarrow \text{adaptat: } \alpha(n, k) = \begin{cases} 1 & |g(n, k) - \hat{f}_b(n, k)| > \varepsilon \\ \alpha & |g(n, k) - \hat{f}_b(n, k)| \leq \varepsilon \end{cases}$$

> pentru zone cu mișcare (\hat{f}_{b1}) sau pentru care estimarea mișcării este eronată (\hat{f}_{b2}) = diferență importantă ⇒ OFF;

> altfel, pentru zone staționare = diferență mică ⇒ pondereare cu α ;

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

$$\hat{f}(n, k) = \hat{f}_b(n, k) + \alpha(n, k) \cdot [g(n, k) - \hat{f}_b(n, k)]$$

B. alegerea lui $\alpha(n, k)$ (continuare):

→ mai fină este adaptarea în sensul LMMSE (Locally Linear Minimal Mean Squared Error):

> principiu: imaginea filtrată = combinație liniară a imaginii înregistrate (posibil degradată) și a imaginii mediate (mediere aritmetică în fiecare pixel pe o anumită vecinătate):

$$\hat{f} = \alpha \cdot \bar{g} + (1 - \alpha) \cdot g \quad \text{unde } \hat{f} \text{ este imaginea filtrată, } g \text{ este imaginea inițială, } \bar{g} \text{ este imaginea mediată } \alpha \text{ un coeficient de reglaj.}$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LMMSE (Locally Linear Minimal Mean Squared Error):

$$\hat{f} = \alpha \cdot \bar{g} + (1 - \alpha) \cdot g$$

> în ipoteza de zgomot alb obținem:

$$g = f + w \quad (f \text{ cadrul neperturbat ideal iar } w \text{ zgomot alb})$$

$$\bar{w} = 0, \quad f \cdot \bar{w} = \bar{f} \cdot \bar{w} = 0$$

$$\Rightarrow \hat{f} = \alpha \cdot (f + w) + (1 - \alpha) \cdot (f + w)$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LMMSE (Locally Linear Minimal Mean Squared Error):

$$\hat{f} = \alpha \cdot (\bar{f} + w) + (1 - \alpha) \cdot (f + w)$$

$$\hat{f} = \alpha \cdot \bar{f} + (1 - \alpha) \cdot (f + w)$$

> eroarea obținută în urma filtrării este:

$$\varepsilon = f - \hat{f} = f - \alpha \cdot \bar{f} - (1 - \alpha) \cdot (f + w)$$

$$= \alpha \cdot (f - \bar{f}) - (1 - \alpha) \cdot w$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LMMSE (Locally Linear Minimal Mean Squared Error):

$$\overline{\varepsilon^2} = \overline{(\alpha \cdot (f - \bar{f}) - (1 - \alpha) \cdot w)^2}$$

$$\overline{\varepsilon^2} = \alpha^2 \cdot \overline{(f - \bar{f})^2} + (1 - \alpha)^2 \cdot \overline{w^2} - 2 \cdot \alpha \cdot (1 - \alpha) \cdot \overline{(f - \bar{f}) \cdot w} = 0$$

dispersii locale

$$\overline{\varepsilon^2} = \alpha^2 \cdot \sigma_f^2 + (1 - \alpha)^2 \cdot \sigma_w^2$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LMMSE (Locally Linear Minimal Mean Squared Error):

$$\overline{\varepsilon^2} = \alpha^2 \cdot \sigma_f^2 + (1 - \alpha)^2 \cdot \sigma_w^2$$

> minimizăm eroarea pătratică medie funcție de α , astfel:

$$\frac{\partial \overline{\varepsilon^2}}{\partial \alpha} = 0 \Rightarrow \frac{\partial \overline{\varepsilon^2}}{\partial \alpha} = 2 \cdot \alpha \cdot \sigma_f^2 - 2 \cdot (1 - \alpha) \cdot \sigma_w^2 = 0$$

$$\Rightarrow \alpha = \frac{\sigma_w^2}{\sigma_f^2 + \sigma_w^2}$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LMMSE (Locally Linear Minimal Mean Squared Error):

$$\alpha = \frac{\sigma_w^2}{\sigma_f^2 + \sigma_w^2}$$

> imaginea f nu este cunoscută, astfel exprimăm pe α funcție de g , astfel:

$$\left. \begin{aligned} g &= f + w \\ \sigma_g^2 &= \sigma_f^2 + \sigma_w^2 \end{aligned} \right\} \Rightarrow \alpha = \frac{\sigma_w^2}{\sigma_g^2}$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LLMMSE (Locally Linear Minimal Mean Squared Error):

$$\alpha = \frac{\sigma_w^2}{\sigma_g^2} = \frac{\sigma_w^2}{\sigma_f^2 + \sigma_w^2}$$

$$\hat{f} = \alpha \cdot \bar{f} + (1 - \alpha)(f + w)$$

$$\Rightarrow \hat{f}_{optimal} = \frac{\sigma_w^2}{\sigma_g^2} \cdot \bar{f} + \left(1 - \frac{\sigma_w^2}{\sigma_g^2}\right) \cdot g$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

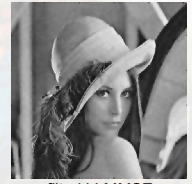
→ LLMMSE (Locally Linear Minimal Mean Squared Error): exemplu imagine statică,



imaginea inițială



imaginea cu zgomot gaussian (dispersie 10)



filtrul LLMMSE (5x5, dispersie=20)



filtrul LLMMSE (5x5, dispersie=10)

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

→ LLMMSE (Locally Linear Minimal Mean Squared Error):

$$\hat{f}_{optimal} = \frac{\sigma_w^2}{\sigma_g^2} \cdot \bar{f} + \left(1 - \frac{\sigma_w^2}{\sigma_g^2}\right) \cdot g$$

B. alegerea lui $\alpha(n,k)$:

→ în acest sens alegem:

$$\alpha(n,k) = \max\left(1 - \frac{\sigma_w^2}{\sigma_{g(n,k)}^2}, 0\right) \text{ unde } \sigma_{g(n,k)}^2 \text{ este un estimat al dispersiei în vecinătatea lui } n \text{ la momentul } k.$$

Filtrarea spațio-temporală a zgomotului

↳ filtre liniare

filtre recursive temporale:

B. alegerea lui $\alpha(n,k)$: $\alpha(n,k) = \max\left(1 - \frac{\sigma_w^2}{\sigma_{g(n,k)}^2}, 0\right)$

→ dacă varianța lui g este importantă = mișcare importantă, estimare mișcare incorectă, atunci OFF:

$$\alpha(n,k) = \max\left(1 - \frac{\sigma_w^2}{\sigma_{g(n,k)}^2}, 0\right) \approx 1$$

→ dacă varianța lui g este comparabilă cu cea a zgomotului = informația furnizată nu este utilă, atunci fără predicție:

$$\alpha(n,k) = \max(\rightarrow 0, 0) \approx 0 \Rightarrow \hat{f}(n,k) = \hat{f}_b(n,k)$$

> M2. Filtrare spațio-temporală [Filtre statistice de ordine]

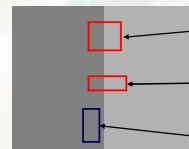
Filtrarea spațio-temporală a zgomotului

↳ filtre statistice de ordine

filtrare neliniară ce se bazează pe ordonarea datelor afectate de zgomot dintr-o vecinătate spațio-temporală,

- generalizarea noțiunii de filtrare de ordonare,

→ de regulă aplicate direcțional pentru a nu afecta contururile și astfel obiectele în mișcare;



frontieră

fereastră de filtrare izotropă, strică conturul

fereastră de filtrare liniară perpendiculară pe contur, strică conturul

fereastră de filtrare liniară paralelă cu conturul, nu strică conturul

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine

- ieșirea unui filtru de ordine de ordin j (rank-order filter) este dată de statistica de ordine de ordin j a valorilor din vecinătatea considerată (fereastra de prelucrare):

$$\text{rank}_j \{A_1, A_2, \dots, A_K\} = A_{(j)}, j = 1, \dots, K$$

unde A_1, \dots, A_K reprezintă valorile prelucrate, K reprezintă numărul de valori din vecinătate, j este ordinul filtrului de ordine iar (j) reprezintă poziția în vectorul ordonat crescător.

$$A_{(1)} < A_{(2)} < \dots < \boxed{A_{(j)}} < \dots < A_{(K)}$$

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine (continuare)

$$\text{rank}_j \{A_1, A_2, \dots, A_K\} = A_{(j)}, j = 1, \dots, K$$

$$\begin{cases} > \text{dacă } j=(K+1)/2 \text{ (K=impar)} & \rightarrow \text{filtrare mediană} \\ > \text{dacă } j=1 & \rightarrow \text{filtru min} \\ > \text{dacă } j=K & \rightarrow \text{filtru max} \end{cases}$$

→ câteva proprietăți:

$$\text{rank}_j \{x_1 + y_1, \dots, x_K + y_K\} \neq$$

$$\text{rank}_j \{x_1, \dots, x_K\} + \text{rank}_j \{y_1, \dots, y_K\}$$

→ neliniar,

$$\text{rank}_j \{g(x_1), \dots, g(x_K)\} = g(\text{rank}_j \{x_1, \dots, x_K\})$$

→ comută cu funcții monotone $g()$

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine (continuare)

→ câteva proprietăți (continuare):

$$\text{rank}_j \{\alpha \cdot x_1 + \beta, \dots, \alpha \cdot x_K + \beta\} = \alpha \cdot \text{rank}_j \{x_1, \dots, x_K\} + \beta$$

→ comută cu operațiile liniare

- exemple:

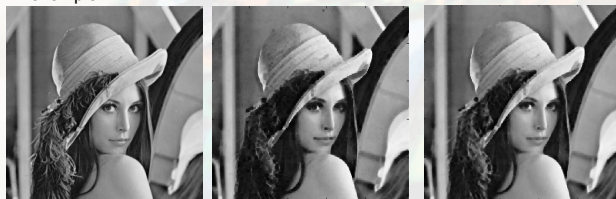


image inițială

$\text{rank}_1 = \min (3 \times 3)$

$\text{rank}_3 = \max (3 \times 3)$

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine (continuare)

- exemple (continuare):



image inițială

$\text{rank}_5 = \text{median} (3 \times 3)$

$\text{rank}_9 = \max (3 \times 3)$

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine (continuare)

- acum și cu zgomot impulsiv:

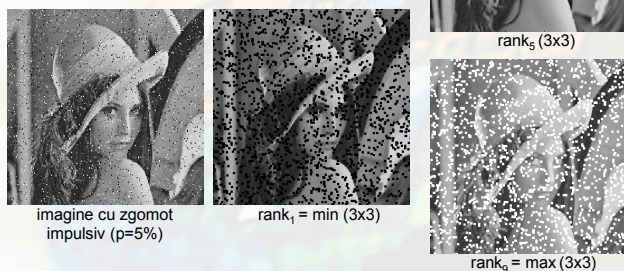


image cu zgomot impulsiv ($p=5\%$)

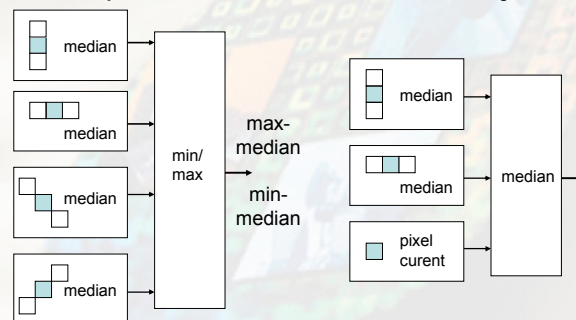
$\text{rank}_1 = \min (3 \times 3)$

$\text{rank}_9 = \max (3 \times 3)$

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine, multietaj

> **multietaj** = o succesiune de filtre de ordine de diferite ranguri:



Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine, multietaj (continuare)

- exemple:



Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine:** la nivel de imagine, multietaj (continuare)

- exemple și cu zgomot impulsiv:



Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine**

[R.L. Lagendijk, J. Biemond, A. Rares, M.J.T. Reinders, Video Enhancement and Restoration, 2009]

→ forma generală în cazul spațio-temporal:

$$\hat{f}(n, k) = \sum_{r=1}^{|S|} h_{(r)}(n, k) \cdot g_{(r)}(n, k)$$

unde $g_{(r)}(n, k)$ reprezintă intensitățile pixelilor ordonate (de rang r), n fiind coordonatele spațiale iar k indicele temporal; S reprezintă fereastra spațio-temporală aleasă în vecinătatea lui n (număr valori = $|S|$).

→ median temporal (poate fi aplicat și cu compensarea mișcării):

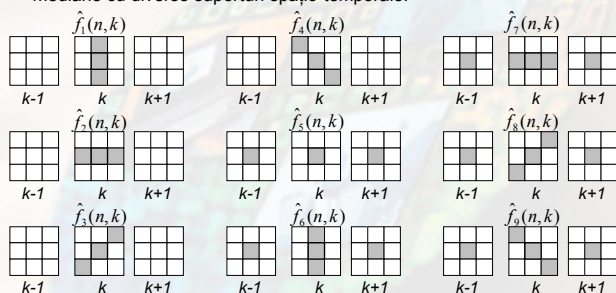
$$\hat{f}(n, k) = \text{median}\{g(n, k-1), g(n, k), g(n, k+1)\}$$

eficient pentru zgomot de tip "shot noise" = zgomotul cauzat de variația numărului de fotoni recepționați la un anumit nivel de expunere (independent, distribuție Poisson).

Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine** (continuare)

→ filtrul median multietaj MMF: sunt combinate ieșirile mai multor filtre mediane cu diverse suporturi spațio-temporale:



Filtrarea spațio-temporală a zgomotului

↳ **filtre statistice de ordine** (continuare)

→ filtrul median multietaj MMF (continuare):

$$\hat{f}(n, k) = \text{median}\left(g(n, k), \max\{\hat{f}_1(n, k), \dots, \hat{f}_9(n, k)\}, \min\{\hat{f}_1(n, k), \dots, \hat{f}_9(n, k)\}\right)$$

> cu toate că nu include estimarea mișcării, reduce artefactele de-a lungul conturilor obiectelor în mișcare datorită orientării filtrării.

> se poate aplica și de-a lungul traiectoriei de mișcare (se orientează ferestrele de-a lungul acesteia).

> M2. Filtrare spațio-temporală
[Filtre multirezoluție]

Filtrarea spațio-temporală a zgomotului

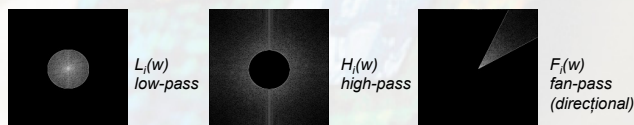
↳ filtre multirezoluție

> *ideea de bază*: descompunerea imaginilor în benzi de rezoluții spațiale, temporale și orientări diferite face ca energia semnalului să fie concentrată într-un anumit număr de benzi, în timp ce zgomotul este prezent în toate benzile;

→ componentele mici din toate benzile (~zgomot) = 0, componentele mari rămân relativ neafectate;

1. descompunere piramidală la nivel spațial:

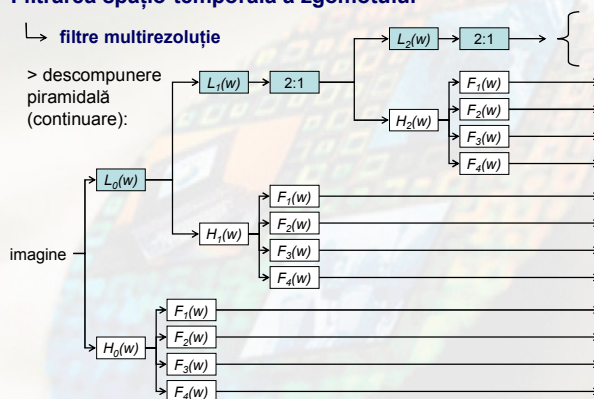
[E.P. Simoncelli, W.T. Freeman, E.H. Adelson, D.J. Heeger, Shiftable Multiscale Transform, 1992]



Filtrarea spațio-temporală a zgomotului

↳ filtre multirezoluție

> descompunere piramidală (continuare):



Filtrarea spațio-temporală a zgomotului

↳ filtre multirezoluție

> introducerea informației de mișcare:

→ dacă aplicăm compensarea mișcării pentru cadrele în intervalul $[k-K; k+K]$ (fereastră temporală) atunci (teoretic):

$$f(k) \approx f(k-1) \approx \dots f(k-K)$$

$$f(k) \approx f(k+1) \approx \dots f(k+K)$$

→ diferențele dintre $g(k)$, $g(k+1)$, ... sunt date doar de zgomot!

→ descompunerea piramidală a acestor imagini va fi de asemenea identică cu excepția zgomotului!

→ dacă reprezentăm temporal coeficienții unei benzi pentru o anumită scală ⇒ un semnal DC + zgomot,

2. descompunere temporală cu DWT (același principiu).

Filtrarea spațio-temporală a zgomotului

↳ filtre multirezoluție

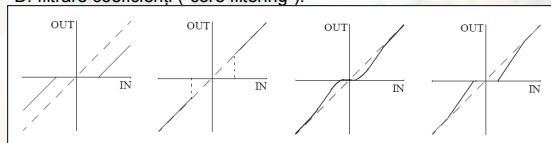
> algoritm de filtrare:

A. pornind de la cadrul curent k → compensarea mișcării (în fereastră),

B. descompunere piramidală pentru fiecare cadru compensat,

C. descompunere wavelet temporală pentru fiecare bandă și frecvență spațială (originea este coeficientul din cadrul k),

D. filtrare coeficienți ("core filtering"):

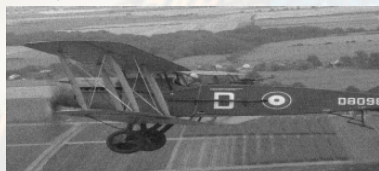


E. reconstrucție inversă: $C^{-1} \rightarrow B^{-1}$.

Filtrarea spațio-temporală a zgomotului

↳ filtre multirezoluție: exemplu

cadru afectat de zgomot



cadru filtrat



Filtrarea spațio-temporală a zgomotului

↳ exemple de filtre de reducere a zgomotului:

[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]

Denoiser name (short name)	Author	Version	Type of application
MSU Denoiser	MSU Video Group	1.6.2	VD
Alparysoft Denoise Filter	Alparysoft	1.0.744.050105	VD
2d cleaner	Jim Casaburi	0.9	VD
Denoiser by "THE FISH"	"THE FISH" (work based on Donald Graft code)	1.0	VD
Dynamic Noise Reduction (DNR)	Steven Don and Avery Lee	22.01.2002	VD
Noise Reduction Suite (NRS)	Antonio Foranna	1.4	VD
Smart Smoother	Klaus Post (work based on Donald Graft code)	2.11	VD
flaXen VHS Filter (VHS)	flaXen	1.0	VD
Video DeNoise	Alexander Tchirkov	2.0	VD (no job)
Neat Video	ABSoft	1.5	VD
Sapphire GrainRemove	GenArts	1.10	AAE
AAE internal grain remover	Adobe Inc.	6.5	AAE

VD = VirtualDub plug-in,

AAE = Adobe After Effects plug-in,

Filtrarea zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare)

> PSNR = Peak Signal-to-Noise Ratio:

$$MSE = \frac{1}{X \times Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (g(x,y) - g'(x,y))^2$$

unde $g()$ este imaginea dimensiune $X \times Y$ pixeli

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX}{MSE} \right)$$

unde MAX reprezintă

Noise types and presets					
	gran noise		normal noise		
	default	for hard noise	default	for hard noise	default
MSU Denoiser	36.62	37.65	36.93	38.45	31.12
Alprosoft Denoise Filter	35.71	29.37	36.26	30.95	30.01
Zil cleaner	37.67	35.60	37.65	36.59	31.51
Denoiser by 'N-E ISP'	36.34	34.42	36.70	35.02	31.04
DNR	36.44	35.25	36.85	35.73	31.14
NRS	36.85	36.07	37.09	37.42	31.22
Smart Smoother	37.73	33.05	38.39	33.81	33.21
VHS	36.89	37.48	36.99	37.68	31.07
Video DeNoise	36.71	33.85	37.55	34.83	33.07
Neat Video	37.47	36.68	36.23	37.75	34.81
Saphire-GraRemove	27.73	28.84	28.55	29.68	28.33
AAE internal	27.31	26.81	29.33	28.70	28.36
Without denoising	36.46	36.46	36.04	36.54	30.94

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

Filtrarea spațio-temporală a zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare):
[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]

secvență originală secvență cu zgomot

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

Filtrarea spațio-temporală a zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare):
[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]

secvență cu zgomot Video DeNoise

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

Filtrarea spațio-temporală a zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare):
[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]

secvență cu zgomot Smart Smoother

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

Filtrarea spațio-temporală a zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare):
[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]

secvență cu zgomot Neat Video

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

Filtrarea spațio-temporală a zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare):
[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]

secvență cu zgomot AAE internal

Analiza și Prelucrarea Secvențelor de Imagini, Ș.I. Bogdan IONESCU

Filtrarea spațio-temporală a zgomotului

→ **exemple de filtre de reducere a zgomotului** (continuare):
[CS MSU Graphics & Media Lab, <http://compression.ru/video/index.htm>]



> M2. Filtrare spațio-temporală [Filtrare artefacte]

Reducerea artefactelor: conturarea blocurilor

surse: A. blocurile DCT, compresie ridicată → apropiere de DC;
B. compensarea mișcării → deplasare blocuri,



> două tipuri de abordări:

- **post filter**: filtrarea este realizată în timpul vizualizării secvenței și astfel nu există o standardizare a acesteia (opțională),
- **loop filter**: filtrarea este realizată în momentul codării secvenței și astfel imaginile filtrate vor fi folosite drept imagini de referință pentru compensarea mișcării (standardizare),

Reducerea artefactelor: conturarea blocurilor

→ **deblocking MPEG-4 AVC**: [P. List, A. Joch, J. Lainema, G. Bjontegaard, M. Karczewicz, Adaptive Deblocking Filter, 2003]

> **avantaje in-loop**:

- garantarea unui anumit nivel de calitate asigurat prin filtrare,
- decodarea nu implică buffer suplimentar pentru filtrare temporală (filtrare la nivel de macro-bloc),
- ameliorare atât a calității obiective cât și subiective.

> **dezavantaje**: complexitatea de calcul (~ 1/3 din complexitatea totală a decodării) + standardizare,

- filtru adaptiv = prelucrări condiționale,
- dimensiunea blocurilor de pixeli mică = 4x4 pixeli.

Reducerea artefactelor: conturarea blocurilor

→ **deblocking MPEG-4 AVC** (continuare):

principiu: prin codarea pe blocuri a imaginii, erorile de codare au tendința să fie mai importante spre marginile blocurilor decât în centru:

122	107	106	111
103	102	101	112
106	100	98	108
118	105	106	120

exemplu de eroare pătatică bloc 4x4

> explicație empirică: valorile din interior dispun de mai multe valori vecine și astfel precizia estimării crește,

→ filtrarea marginilor blocurilor poate conduce la îmbunătățirea calității!

adaptiv în funcție de conținut:

→ **la nivel de frontieră** între blocuri: adaptare la codarea inter/intra, diferențele de mișcare și eroare reziduală,

Reducerea artefactelor: conturarea blocurilor

→ **deblocking MPEG-4 AVC** (continuare):

adaptiv în funcție de conținut:

→ **la nivel de frontieră** între blocuri: pentru fiecare frontieră dintre două blocuri de luminanță de 4x4 pixeli se definește un parametru **Bs** (Boundary-Strength):

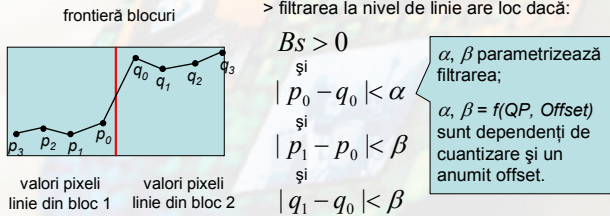
ordinea de evaluare	Block modes and conditions	Bs	
1	One of the blocks is Intra and the edge is a macroblock edge	4	filtrare puternică
2	One of the blocks is Intra	3	
3	One of the blocks has coded residuals	2	
4	Difference of block motion ≥ 1	1	mod standard (Bs afectează gradul maxim de modificare al valorilor)
5	luma sample distance	1	
6	Motion compensation from different reference frames	1	
7	Else	0	fără filtrare

Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC (continuare):

modul de filtrare:

→ la nivel de valori: diferențiere între contururile reale (→ nefiltrate) din imagine și cele artificiale create de compresie,



Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC (continuare):

modul de filtrare:

→ la nivel de slice: adaptare la caracteristicile secvenței,

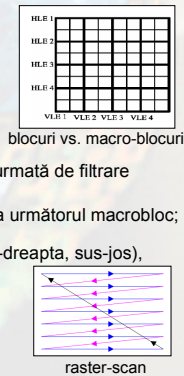
- > prin intermediul $Offset_A$ și $Offset_B$ se ajustează valorile pragurilor α și respectiv β și astfel se reglează gradul de filtrare,
- > aceste valori sunt transmise în antetul slice-ului,
- > optimizare calitate subiectivă:
 - *offset negativ* reduce filtrarea și astfel contribuie la păstrarea acuității micilor detalii spațiale (HD unde artefactele 4x4 sunt puțin vizibile),
 - *offset pozitiv* crește filtrarea și astfel contribuie la îmbunătățirea calității subiective atunci când artefactele sunt încă vizibile (rezoluții mici, netezirea tranzițiilor),

Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC (continuare):

modul de filtrare:

- > "in-place" = valorile modificate sunt folosite în continuare pentru următorii pași de filtrare,
- > filtrare la nivel de macro-bloc,
- > ordine: filtrare orizontală a frontierelor verticale urmată de filtrare verticală a frontierelor orizontale,
 - după ce se filtrează ambele direcții se trece la următorul macrobloc;
- > macro-blocurile sunt scanate secvențial (stânga-dreapta, sus-jos),
 - "raster-scan";
- > două moduri de filtrare funcție de Bs:
 - $Bs = 4$ – filtrare puternică,
 - $Bs = \{1, 2, 3\}$ – filtrare adaptată.



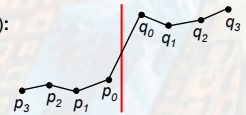
Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC (continuare):

modul de filtrare (continuare):

> Boundary-Strength (Bs) = 1, ..., 3:

- netezire în funcție de 4 valori p_1, p_0, q_0, q_1
 - calcul valori noi p'_0, q'_0
- dacă $|p_2 - p_0| < \beta$ netezire în funcție de 4 valori p_2, p_1, p_0, q_0
 - calcul valoare nouă p'_1
- dacă $|q_2 - q_0| < \beta$ netezire în funcție de 4 valori q_2, q_1, q_0, p_0
 - calcul valoare nouă q'_1

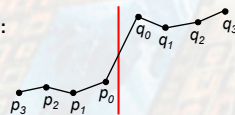


Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC (continuare):

modul de filtrare (continuare):

- > Boundary-Strength (Bs) = 4:
- 1. dacă $|p_2 - p_0| < \beta$ și $|p_0 - q_0| < \alpha / 4$
 - calcul valoare nouă p'_0 în funcție de p_2, p_1, p_0, q_0, q_1 ,
 - calcul valoare nouă p'_1 în funcție de p_2, p_1, p_0, q_0 ,
 - calcul valoare nouă p'_2 în funcție de p_3, p_2, p_1, p_0, q_0 ,
- altfel
 - calcul valoare nouă p'_0 în funcție de p_1, p_0, q_0 .



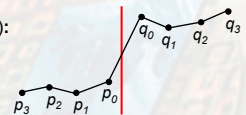
Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC (continuare):

modul de filtrare (continuare):

> Boundary-Strength (Bs) = 4:

2. dacă $|q_2 - q_0| < \beta$ și $|p_0 - q_0| < \alpha / 4$
 - calcul valoare nouă q'_0 în funcție de q_2, q_1, q_0, p_0, p_1 ,
 - calcul valoare nouă q'_1 în funcție de q_2, q_1, q_0, p_0 ,
 - calcul valoare nouă q'_2 în funcție de q_3, q_2, q_1, q_0, p_0 ,
- altfel
 - calcul valoare nouă q'_0 în funcție de q_1, q_0, p_0 .



Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC: exemplu 1



decompresie (original)



decompresie (fără deblocking)

Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC: exemplu 2



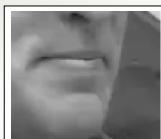
decompresie (original)



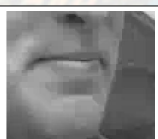
decompresie (fără deblocking)

Reducerea artefactelor: conturarea blocurilor

↳ deblocking MPEG-4 AVC: exemplu 3



cu deblocking



fără deblocking



fără deblocking



cu deblocking

[P. List, A. Joch, J. Lainema, G. Bjontegaard, M. Karczewicz, Adaptive Deblocking Filter, 2003]

> Sfârșit M2. Filtrare spațio-temporală