Concepte și aplicații în Vederea Artificială

Bogdan Alexe
bogdan.alexe@fmi.unibuc.ro

Cursul 3

anul III, Opțional Informatică, semestrul I, 2018-2019

Cursul trecut

- Formarea imaginilor
- Primul proiect
- Diverse modele pentru zgomot în imagini
  - salt and pepper, impuls
  - Gaussian (normal)
- Filtrarea liniară
  - corelație, convoluție
  - filtre: de medie, Gaussian, accentuare
  - aplicatie: imagini hibrid

normal

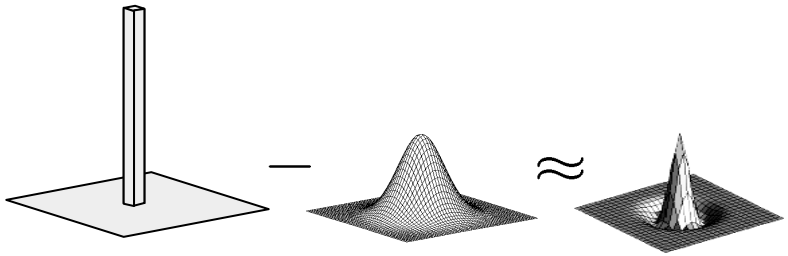
Aplicație: Imagini hibrid

filtru Gaussian
(low pass filter – obține frecvențe joase)

-

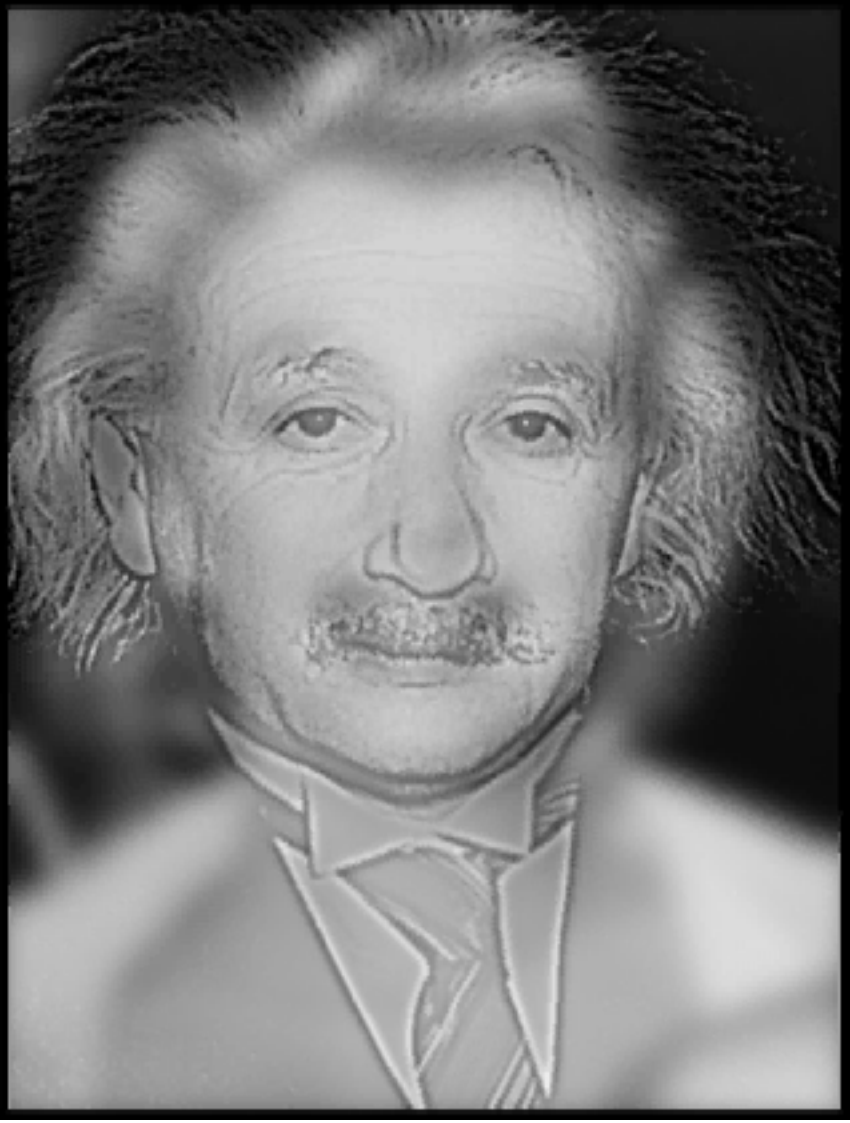
-

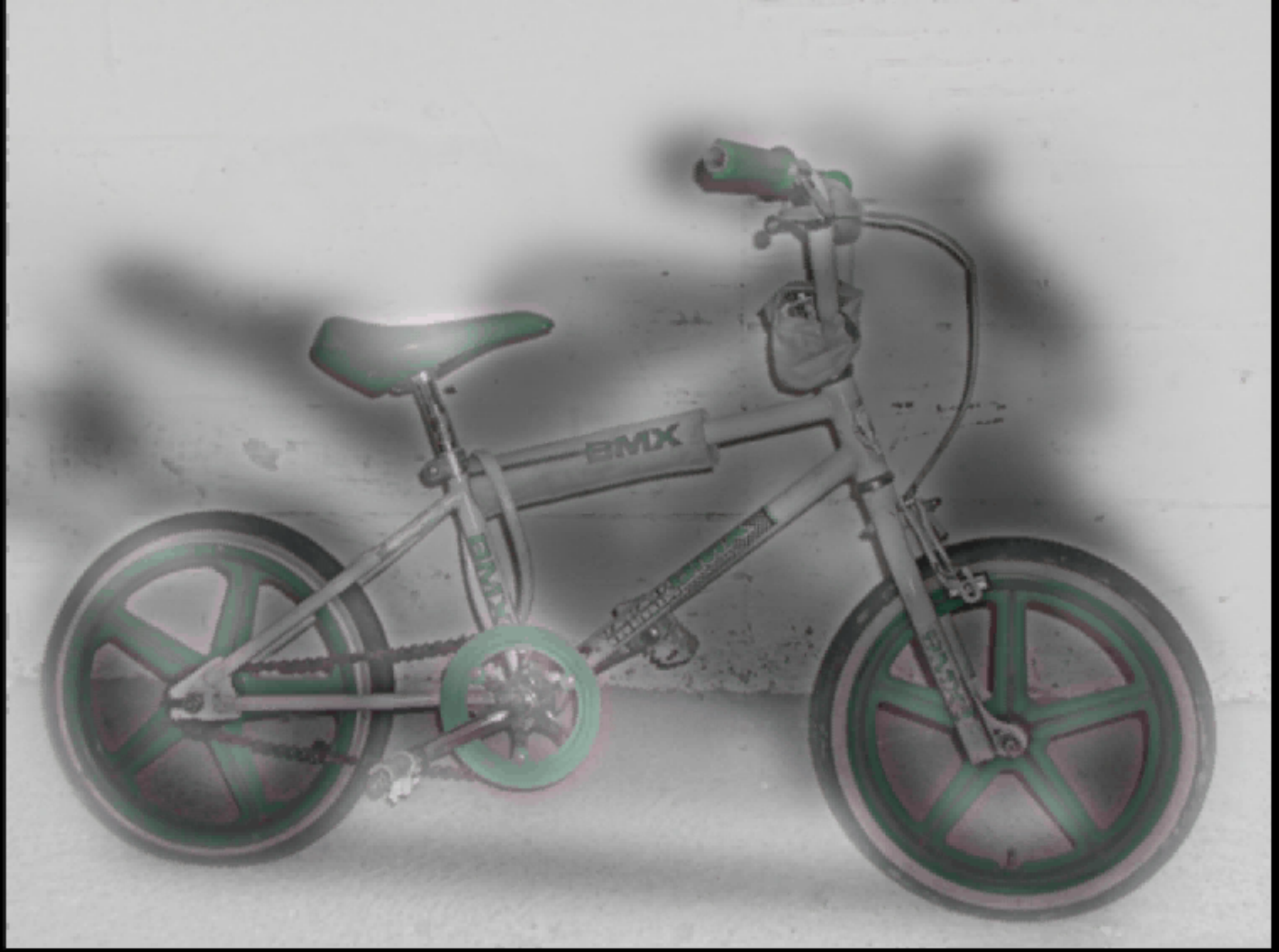
filtru Laplacian
(high pass filter – obține frecvențe înalte)

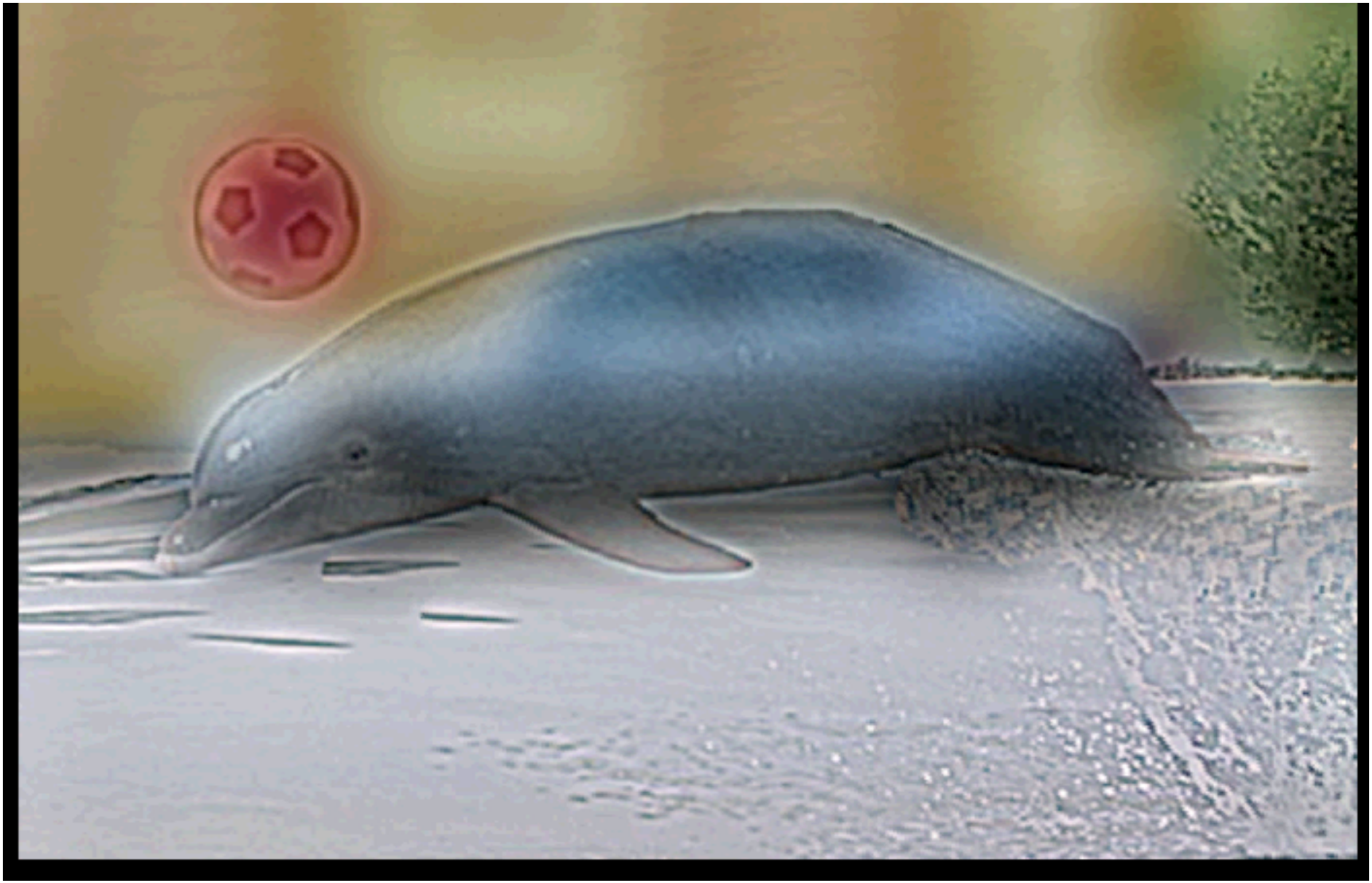


impuls Gaussian Laplacian

A. Oliva, A. Torralba, P.G. Schyns,
"Hybrid Images" SIGGRAPH 2006




BMX



Cursul de azi

- Filtrarea liniară
  - corelație, convoluție
  - filtre: accentuare

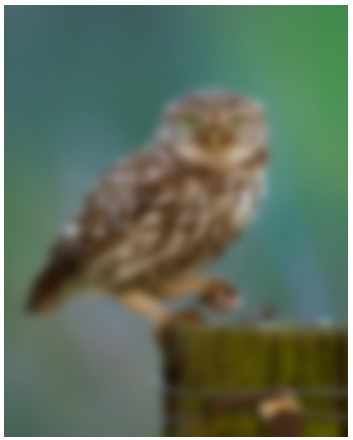
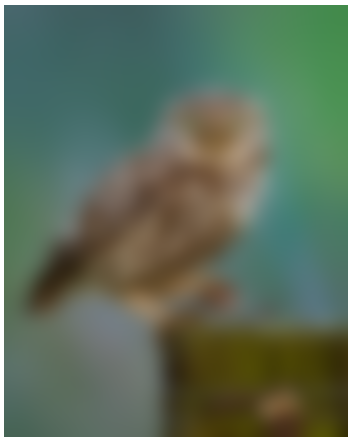
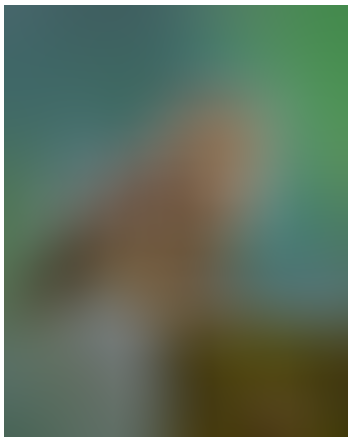
- Filtrarea neliniară
  - filtrul median

- Aplicații:
  - reducerea zgomotului în imagini (filtrul median)
  - găsirea şabloanelor (cursul de azi)
  - redimensionarea imaginilor (cursul de azi)
  - extragerea informației (muchii, textură – cursul de azi + săptămâna viitoare)

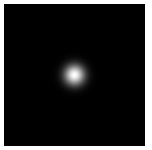
Filtrarea imaginilor color

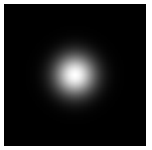
$$I(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix} \Rightarrow F \otimes I = \begin{bmatrix} F \otimes r \\ F \otimes g \\ F \otimes b \end{bmatrix}$$

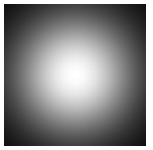



σ = 1 pixel

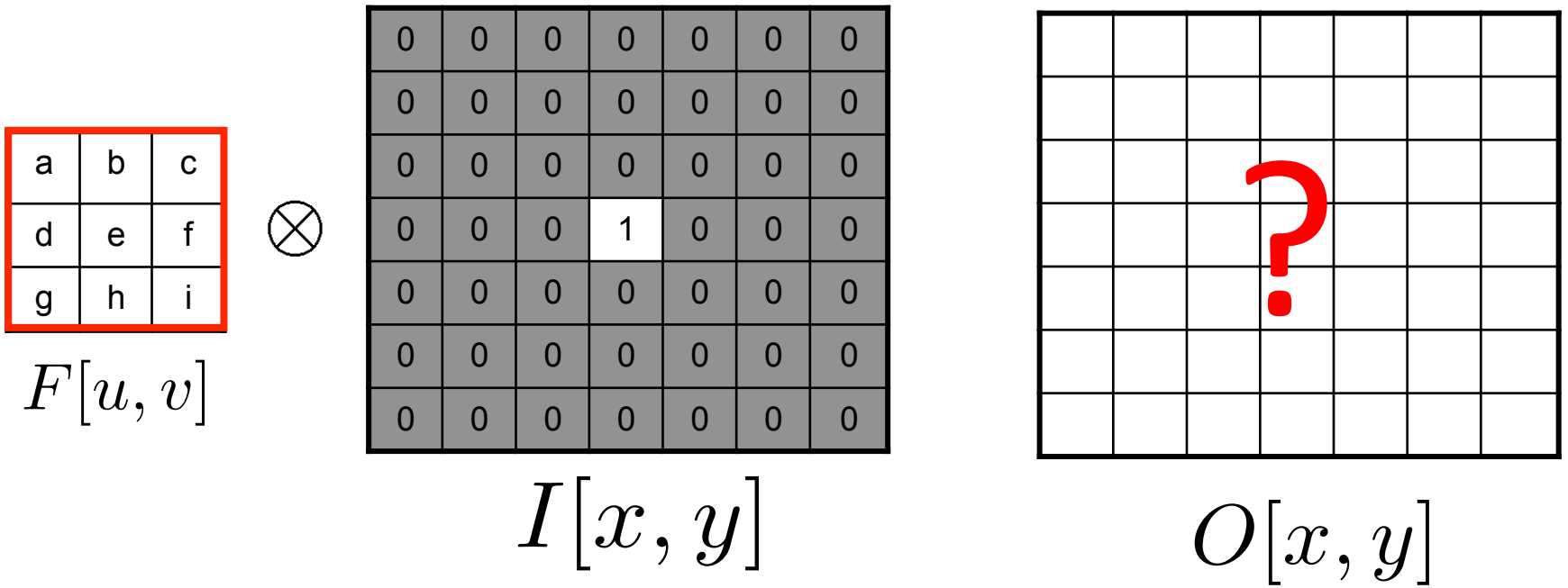

σ = 5 pixels


σ = 10 pixels


σ = 30 pixels

Filtrarea unui semnal impuls

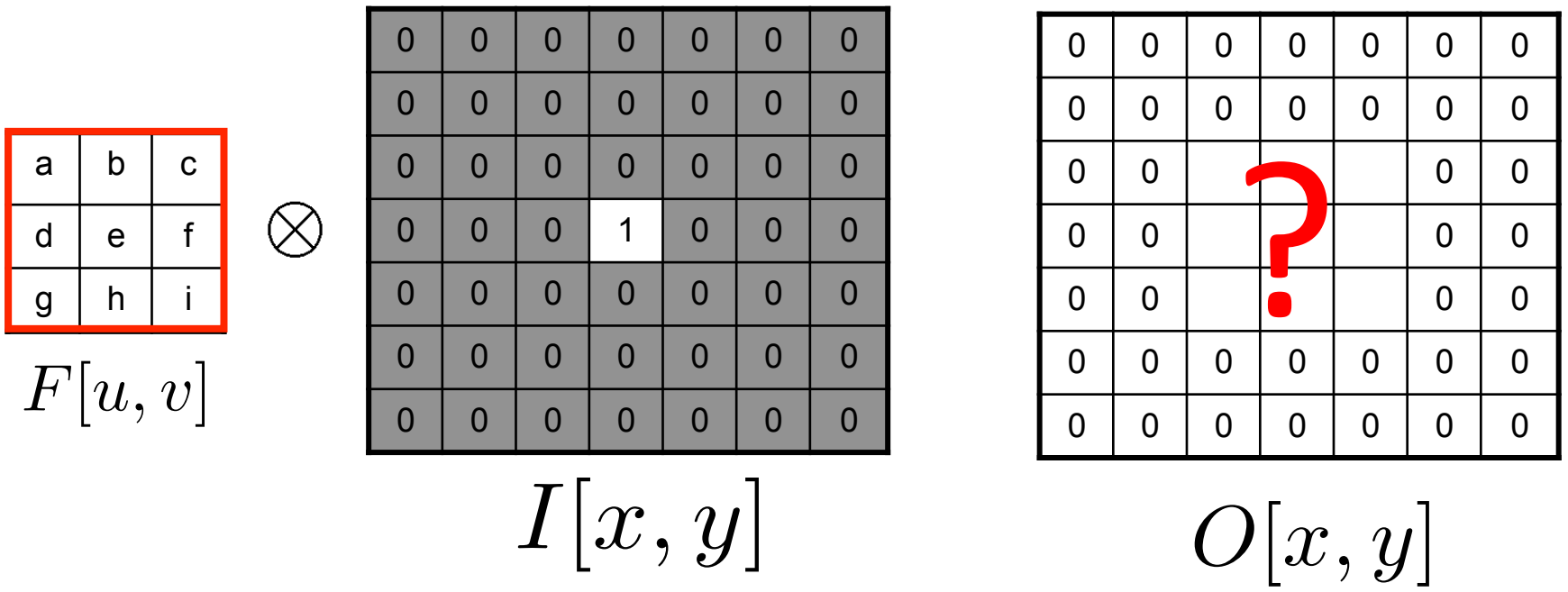
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

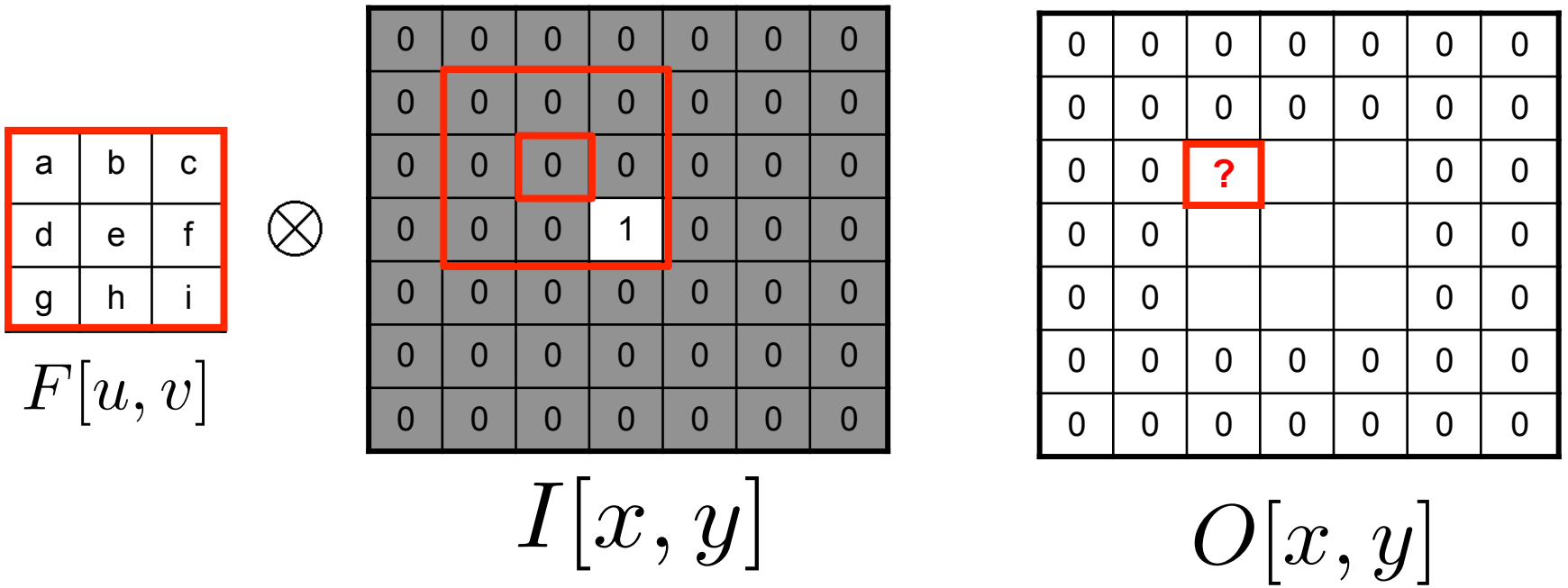
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

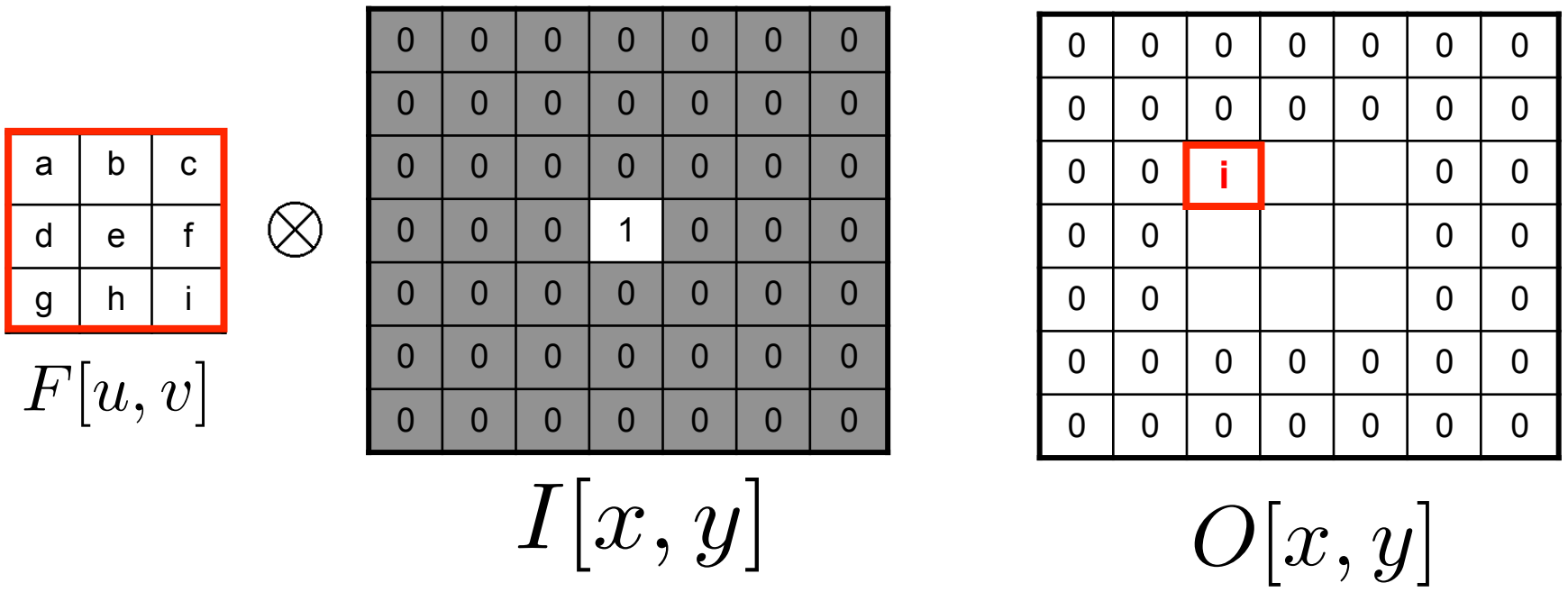
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

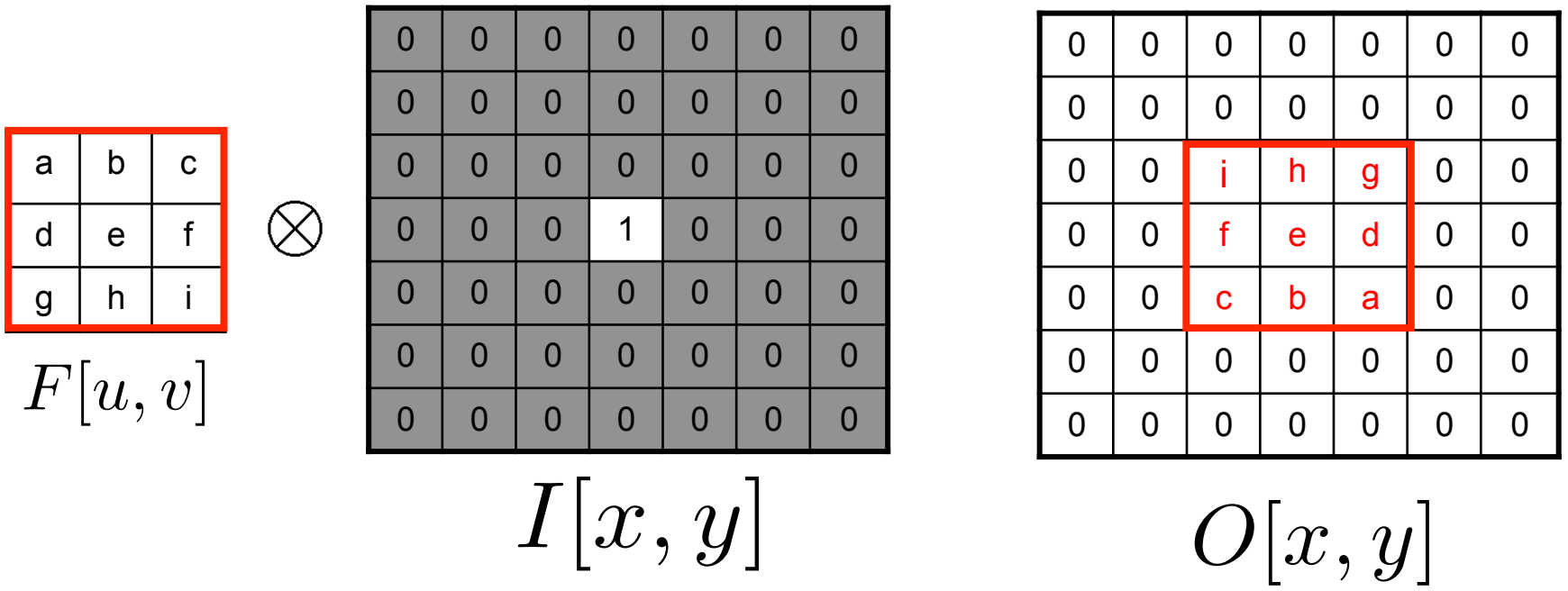
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



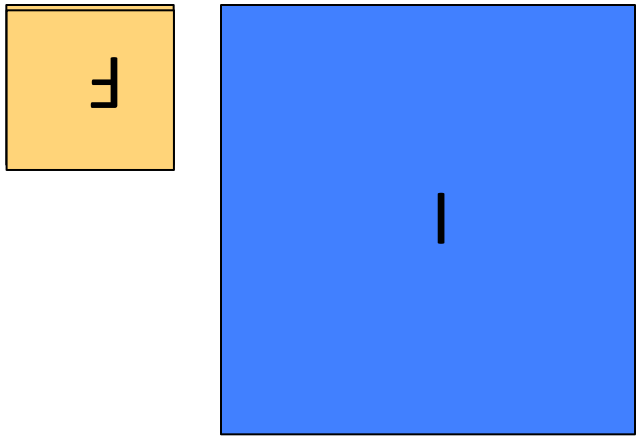
Convoluție

- Convoluție:
  - inversăm filtrul în ambele direcții (jos ⇔ sus, dreapta ⇔ stânga)
  - aplicăm corelația

$$O[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u,v] I[i-u, j-v]$$

$$O = F \star I$$

notație pentru convoluție



- MATLAB: conv2, imfilter

Convoluție vs. corelație

Convoluție

$$O[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u,v] I[i-u, j-v]$$

$$O = F \star I$$

Corelație

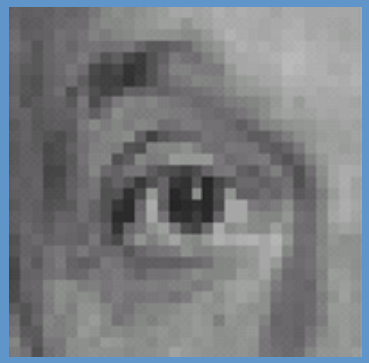
$$O[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u,v] I[i+u, j+v]$$

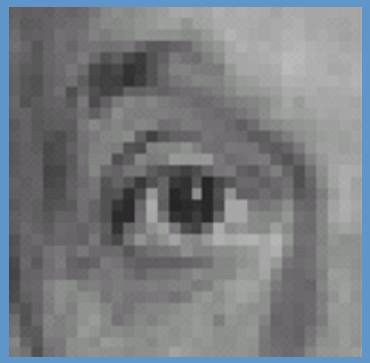
$$O = F \otimes I$$

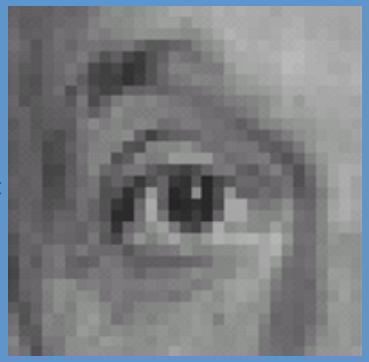
Pentru un filtru normal/de medie, cum diferă output-ul?

Pentru input = un semnal impuls, cum diferă output-ul?

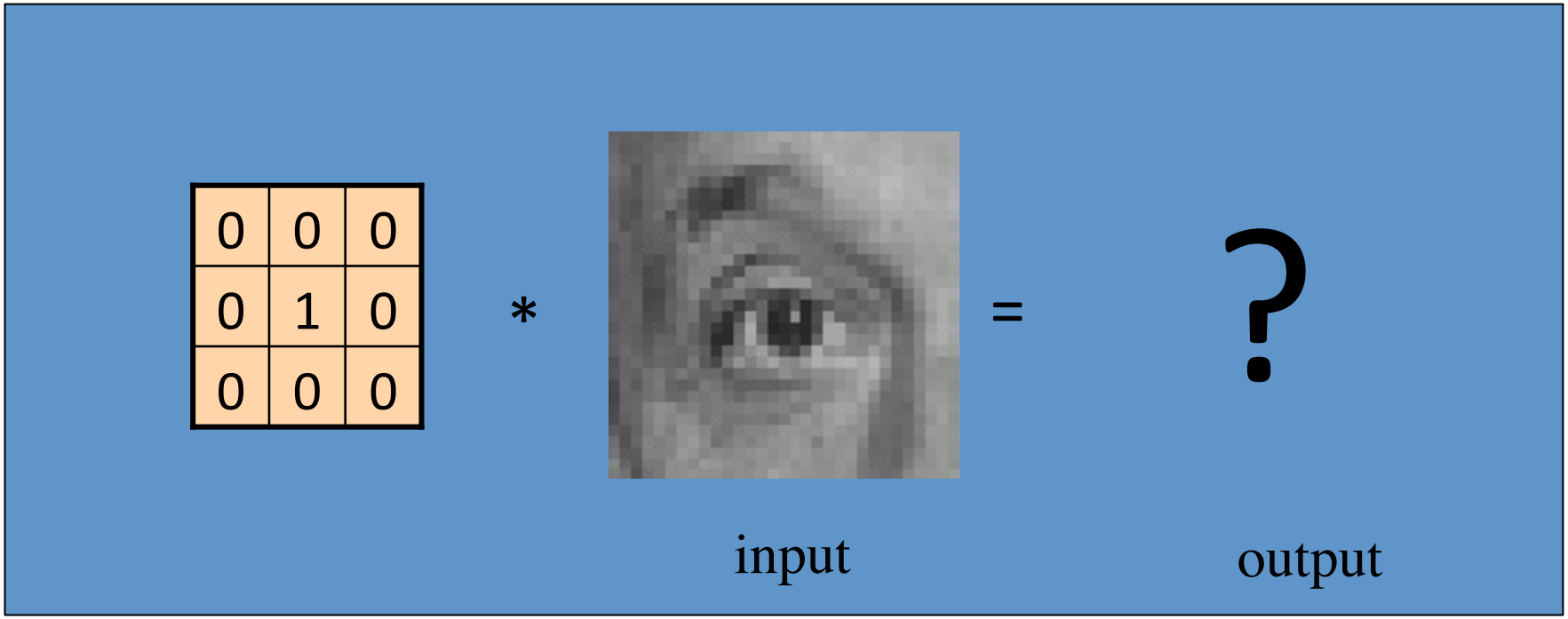
Output = ?

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \text{  } = ?$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \text{  } = ?$$

$$\left[\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right] * \text{  } = ?$$

Filtre liniare

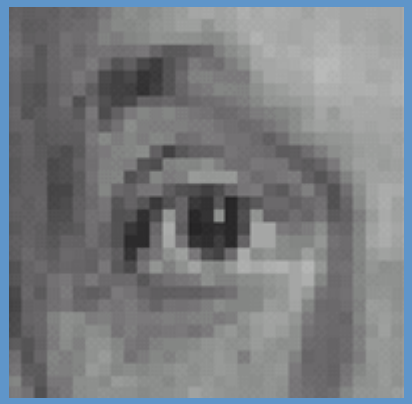


Slide adaptat după David Lowe

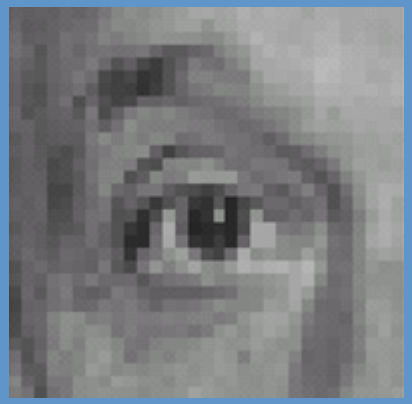
Filtre liniare

0	0	0
0	1	0
0	0	0

*



=

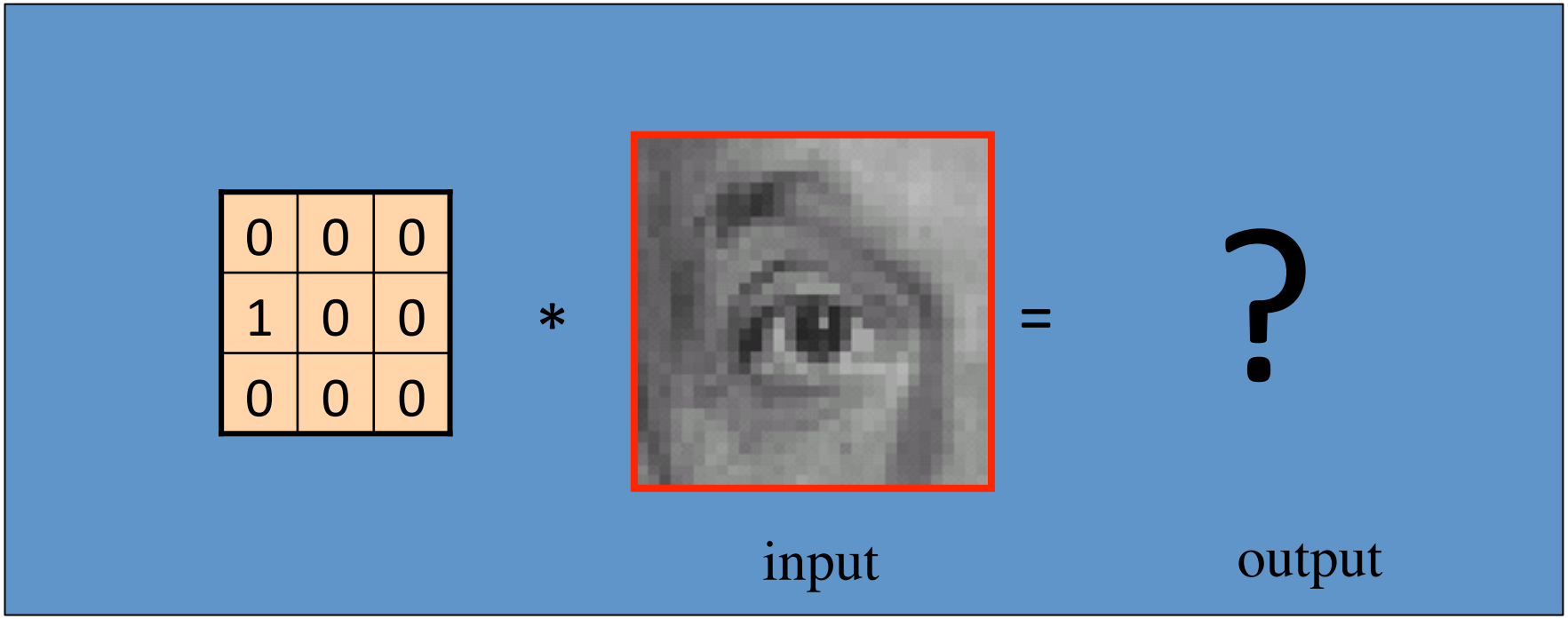


input

output (la fel)

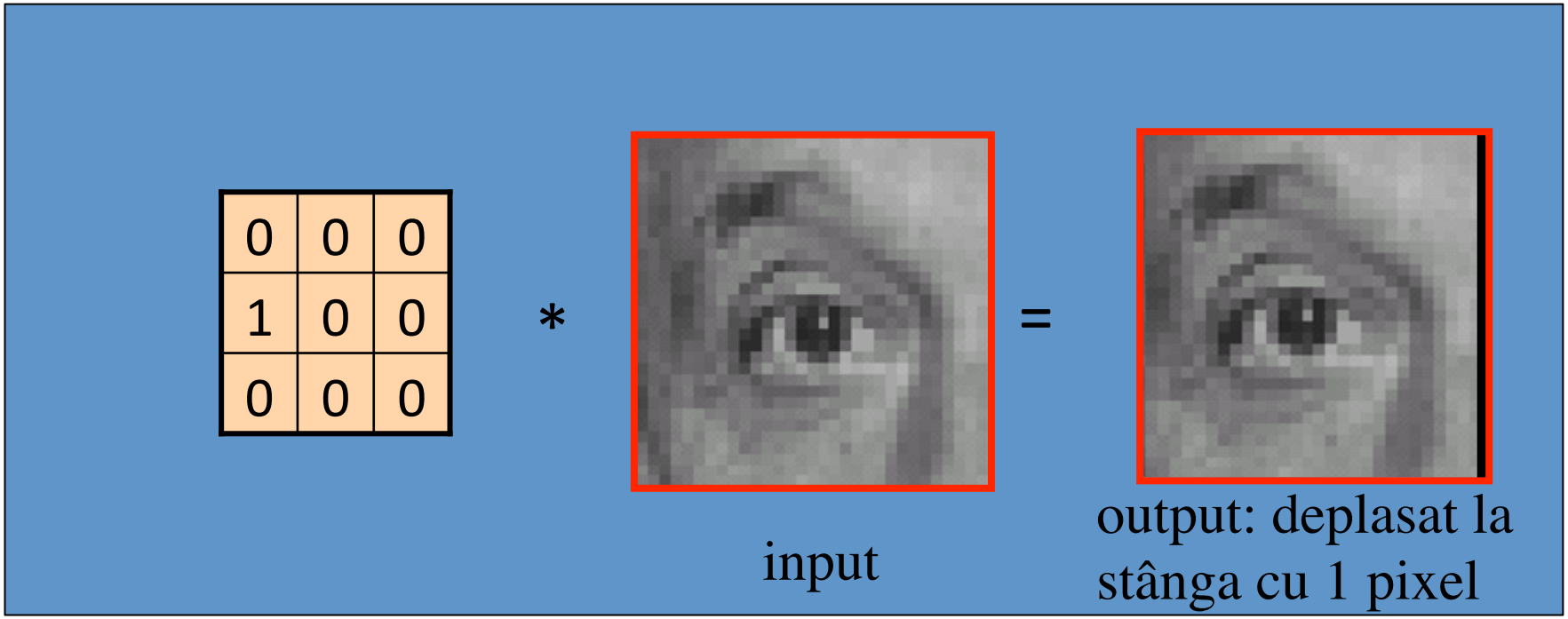
Slide adaptat după David Lowe

Filtre liniare



Slide adaptat după David Lowe

Filtre liniare



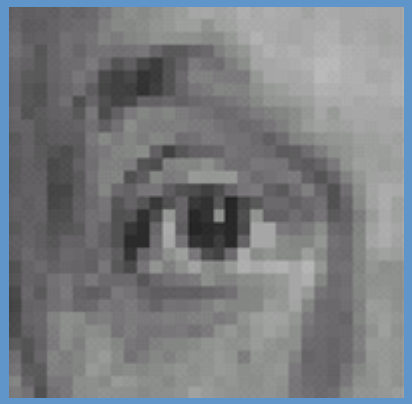
Slide adaptat după David Lowe

Filtre liniare

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

*



=

?

input

output

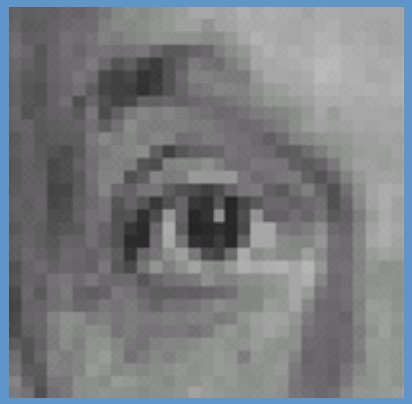
Slide adaptat după David Lowe

Filtre liniare

$\frac{1}{9}$

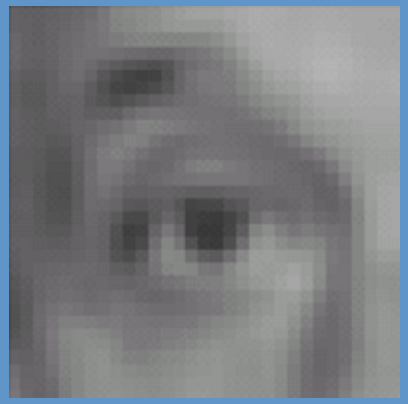
1	1	1
1	1	1
1	1	1

*



input

=



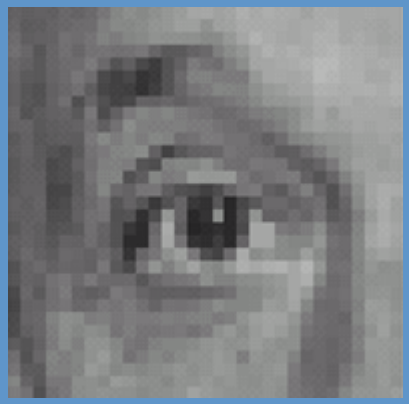
blurat
(cu filtru de medie)

Slide adaptat după David Lowe

Filtre liniare

$$\left[\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} - \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \right] * \text{input} = \text{output}$$

(Suma = 1)



?

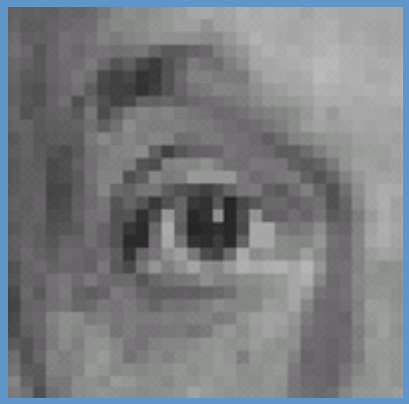
Slide adaptat după David Lowe

Filtre liniare

$\frac{1}{9}$

-1	-1	-1
-1	17	-1
-1	-1	-1

*



=

?

(Suma = 1)

input

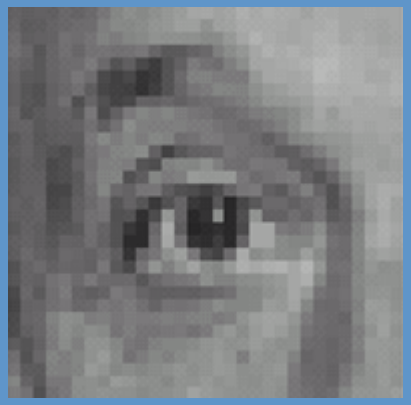
output

Filtre liniare

$\frac{1}{9}$

-1	-1	-1
-1	17	-1
-1	-1	-1

*



=

(Suma = 1)

input

output

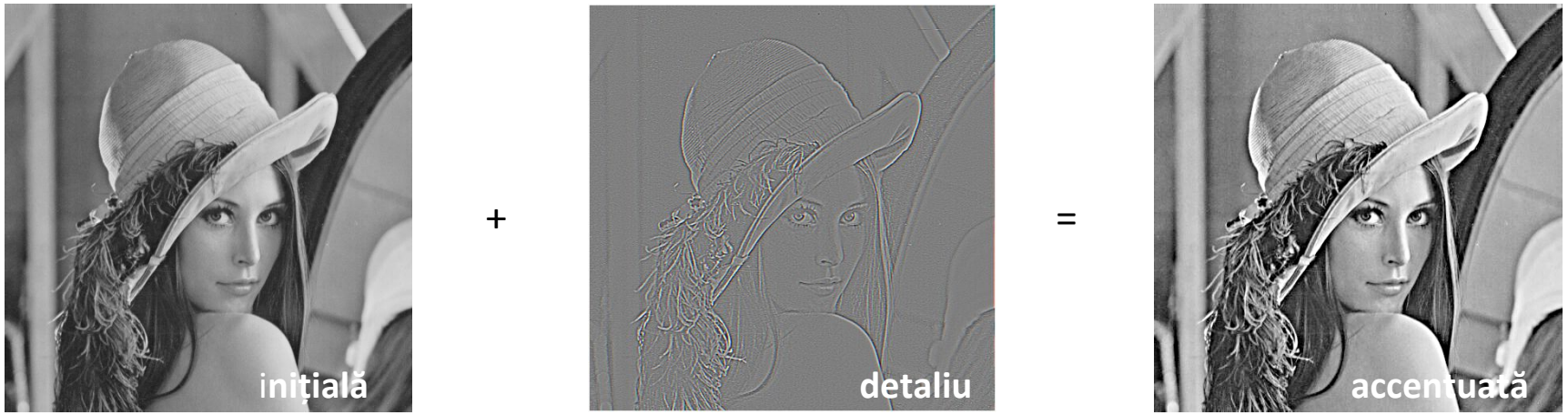
Filtru de accentuare - accentuează diferențele cu ajutorul mediilor locale

Slide adaptat după David Lowe

Filtrul de accentuare

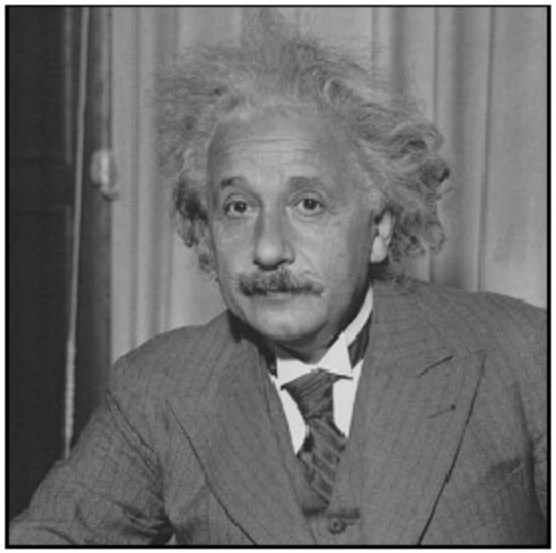
- Ce pierdem în urma blurării unei imagini?

- Adăugam detaliul:

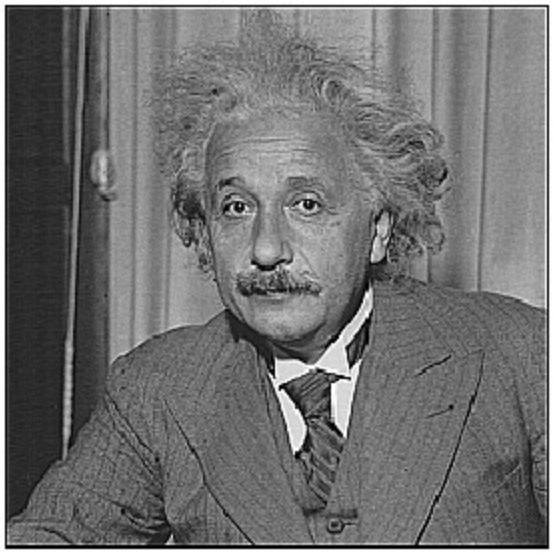


Slide adaptat după S. Lazebnik

Filtrul de accentuare



input



output

Slide adaptat după S. Lazebnik

Reducerea de zgomot

imagine originală

“salt and pepper”

impuls

normal

Reducerea zgomotului “salt and pepper”

Aplicăm un filtru Gaussian de dimensiune n x n

3×3

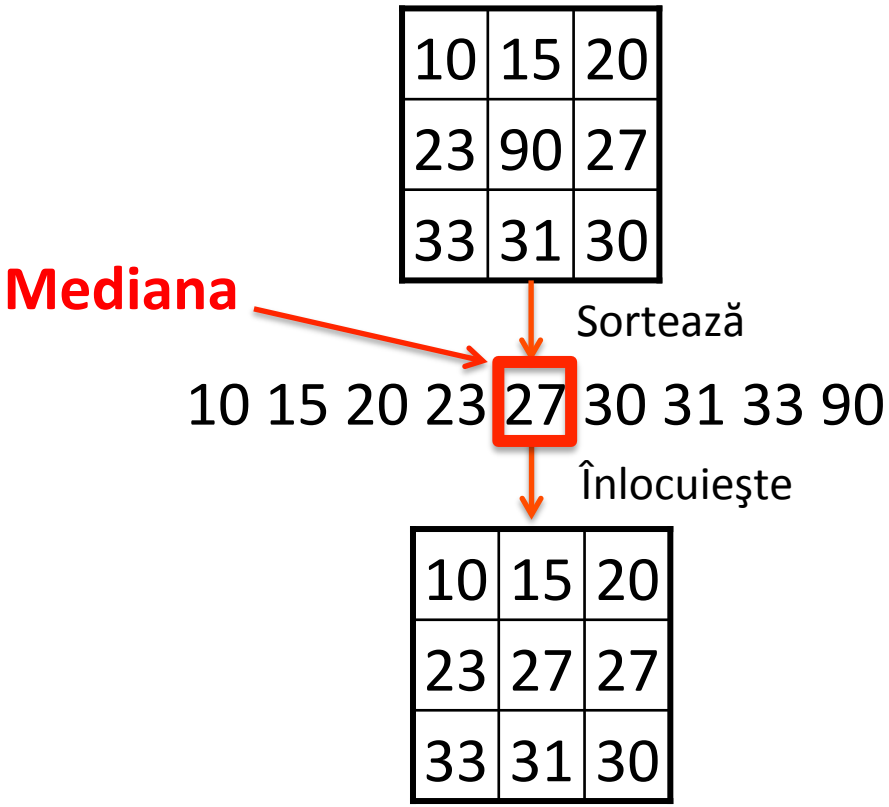
5×5

7×7

Zgomotul se elimină dar imaginea se deteriorează
(blurare + zgomotul se împrăștie la pixelii vecini)

Filtru median

- Un **filtru median** operează asupra unei ferestre prin selectarea intensității mediane



Slide adaptat după K. Grauman

Filtru median

- Un **filtru median** operează asupra unei ferestre prin selectarea intensității mediane

- Este liniar? (median(I_1+I_2) = median(I_1) + median(I_2))

mediana($\begin{bmatrix} 10 & 10 & 10 \\ 10 & 15 & 20 \\ 20 & 20 & 20 \end{bmatrix}$ + $\begin{bmatrix} 10 & 10 & 10 \\ 10 & 5 & 20 \\ 20 & 20 & 20 \end{bmatrix}$) = mediana($\begin{bmatrix} 20 & 20 & 20 \\ 20 & 20 & 40 \\ 40 & 40 & 40 \end{bmatrix}$) = 20

mediana($\begin{bmatrix} 10 & 10 & 10 \\ 10 & 15 & 20 \\ 20 & 20 & 20 \end{bmatrix}$) + mediana($\begin{bmatrix} 10 & 10 & 10 \\ 10 & 5 & 20 \\ 20 & 20 & 20 \end{bmatrix}$) = 15 + 10 = 25

- NU este liniar

Filtrul median vs. filtrul Gaussian

3×3 5×5 7×7

filtru Gaussian



zgomot "salt and peper"

filtru median

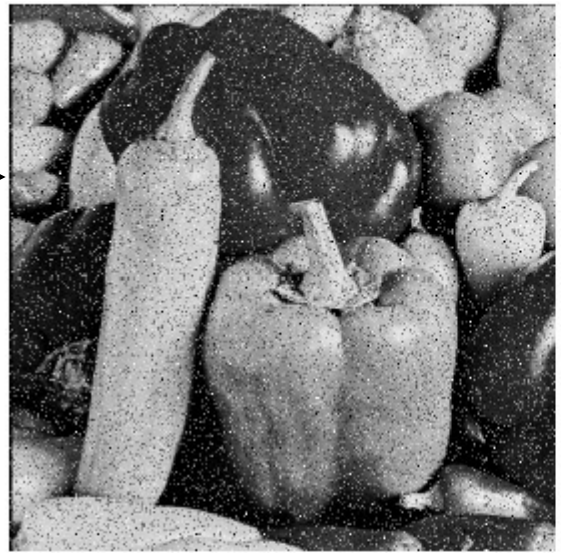


Zgomotul se elimină imaginea cu un filtru median 3 x 3. Pentru vecinătăți mai mari imaginea se deteriorează (se uniformizează).

Filtrul median

zgomot

“salt and peper”



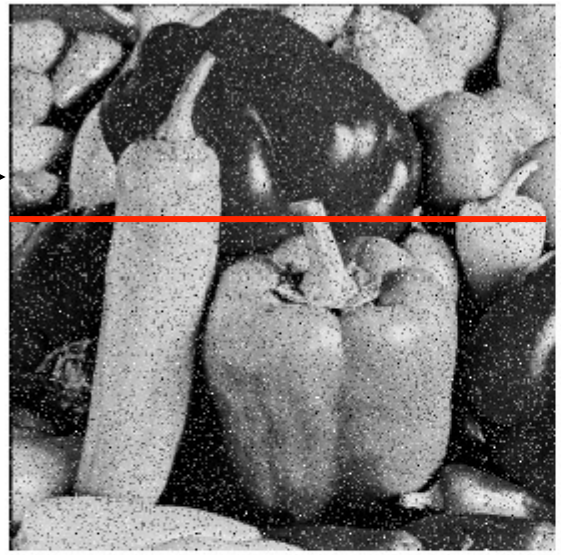
după aplicarea filtrului median

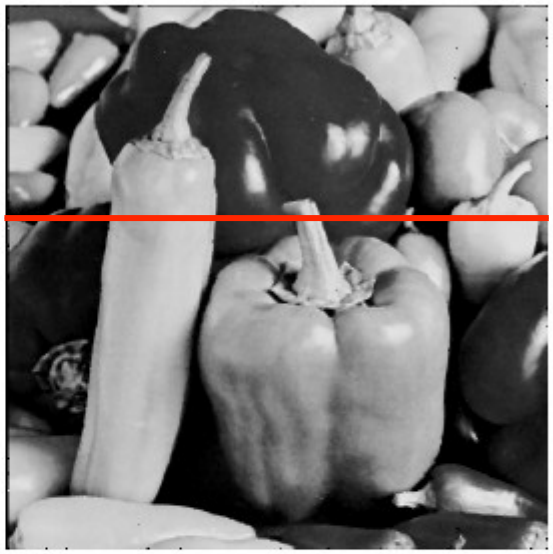
Slide adaptat după M. Herbert

Filtrul median

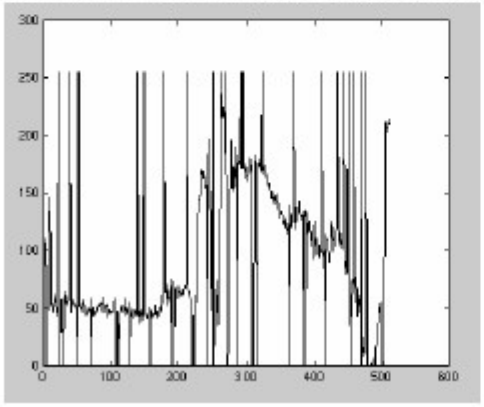
zgomot

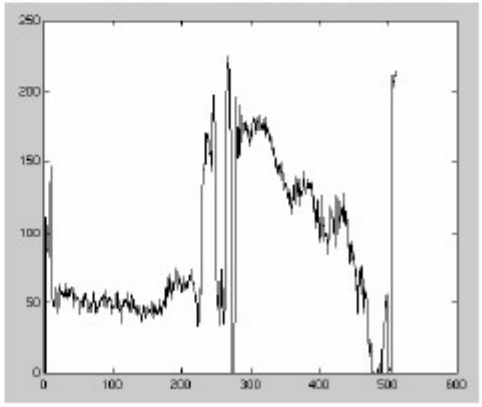
"salt and peper"





după aplicarea filtrului median





Plotăm intensitatea pentru pixelii (de pe linia roșie) din imagini

Matlab: output_im = medfilt2(im, [h w]);

Slide adaptat după M. Herbert

Găsirea şabloanelor (template matching)

Găsirea şabloanelor - exemplu

Vrem să găsim în imaginea alăturată şablonul de mai jos:

şablon
(sens unic)

- cea mai simplă metodă de detectare a obiectelor
- pentru fiecare pixel din imagine, compară şablonul cu fereastra centrată la acel pixel: măsoară cât de bine se aseamănă fereastra cu şablonul (similaritate)

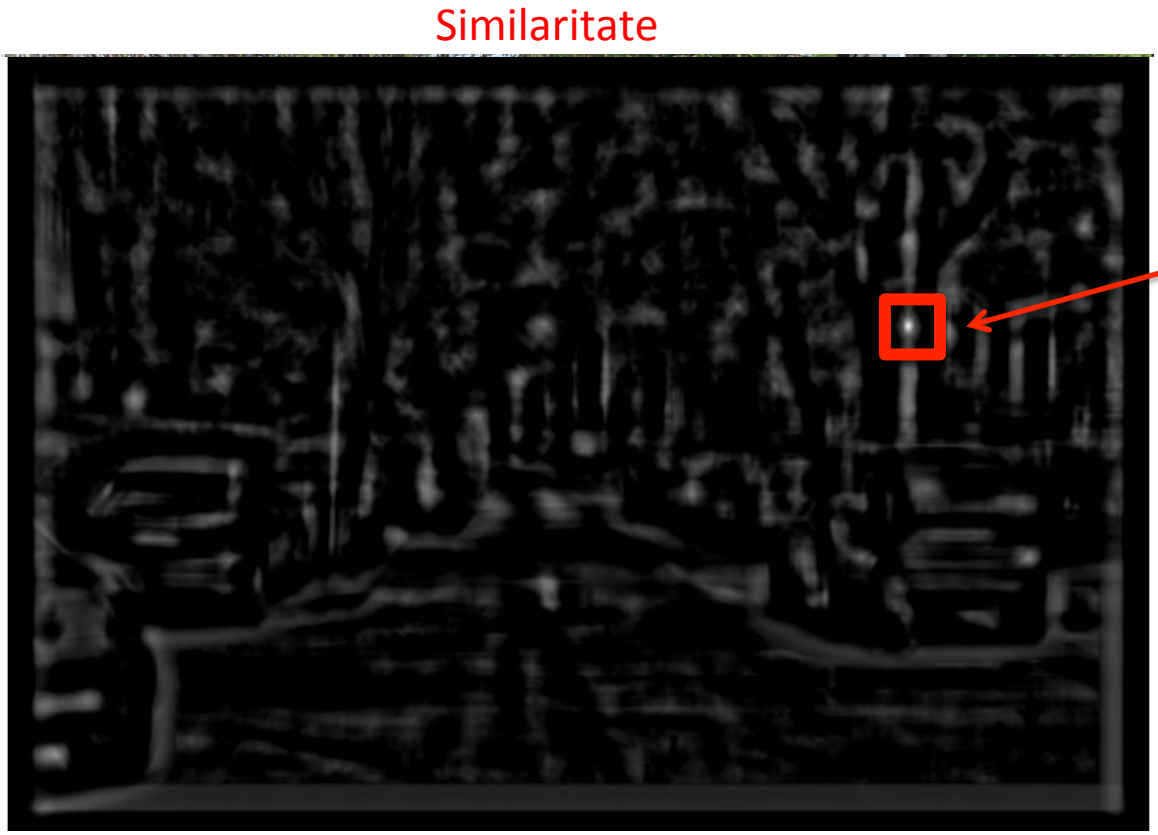
Găsirea şabloanelor - exemplu

Vrem să găsim în imaginea alăturată şablonul de mai jos:

şablon
(sens unic)

- cea mai simplă metodă de detectare a obiectelor
- pentru fiecare pixel din imagine, compară şablonul cu fereastra centrată la acel pixel: măsoară cât de bine se aseamănă fereastra cu şablonul (similaritate)

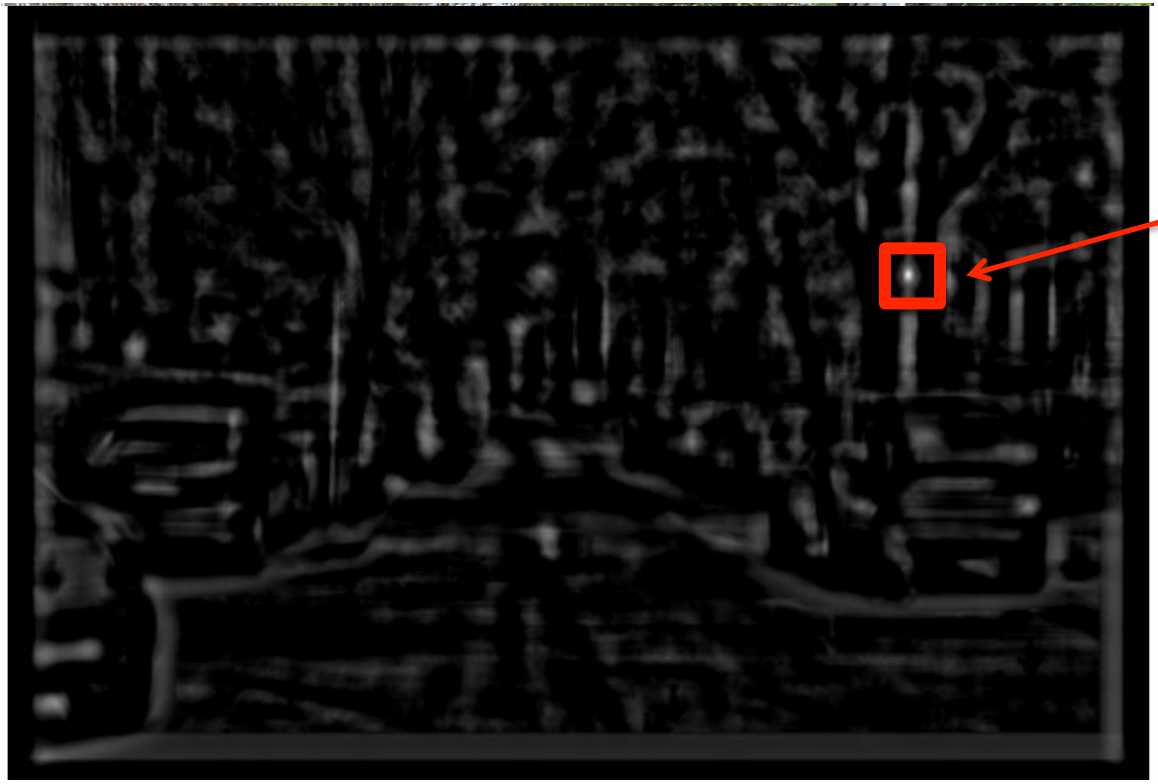
Găsirea şabloanelor - exemplu



- cea mai simplă metodă de detectare a obiectelor
- pentru fiecare pixel din imagine, compară şablonul cu fereastra centrată la acel pixel: măsoară cât de bine se aseamănă fereastra cu șablonul (similaritate)

Găsirea şabloanelor - exemplu

Similaritate



Valoarea cea mai mare a similarității

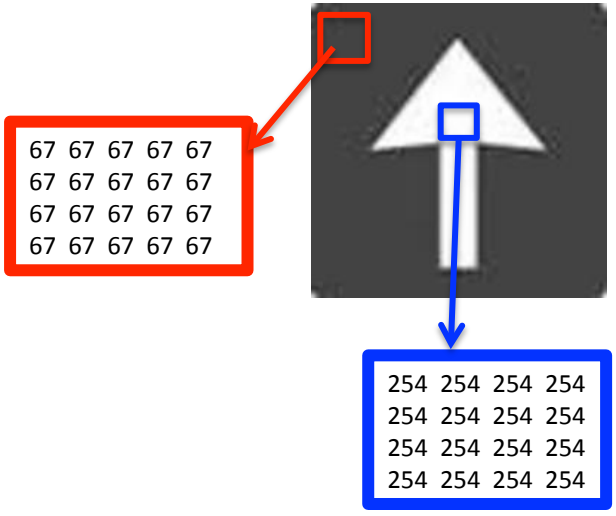
șablon
(sens unic)

- similaritate = imagine filtrată (şablon = filtru)
- o măsură bună pentru similaritatea a două ferestre?
  - corelația?
  - suma pătratelor distanțelor?
  - altceva?

Găsirea şabloanelor cu filtre

- şablon = 

rgb2gray



rgb2gray(img)

Slide adaptat după Derek Hoiem

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- corelație: filtrăm imaginea cu filtrul f_1

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_1(u,v) I(i+u, j+v)$$

I

$$O_1 = f_1 \otimes I$$

Ce rezultă după filtrare?

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- corelație: filtrăm imaginea cu filtrul f_1

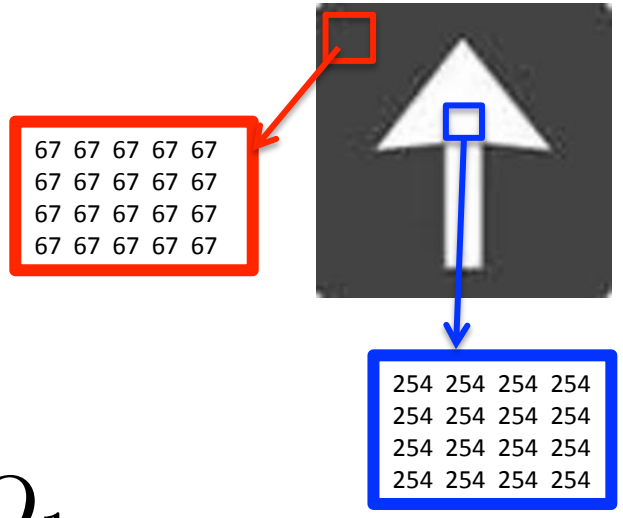
$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_1(u,v) I(i+u, j+v)$$

imagine complet albă

O_1

Explicații?

Filtru ne-normalizat (suma > 1) rezultă în imagine filtrată cu luminozitate crescută



Găsirea şabloanelor cu filtre

- filtrul f_2 = normalizat (suma = 1: 67→ 0.00007, 254→0.00003)
- corelație: filtrăm imaginea cu filtrul f_2

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_2(u,v) I(i+u, j+v)$$

I

$$O_2 = f_2 \otimes I$$

Ce rezultă după filtrare?

Găsirea şabloanelor cu filtre

- filtrul f_2 = ⬆ normalizat (suma = 1: 67→ 0.00007, 254→0.00003)
- corelație: filtrăm imaginea cu filtrul f_2

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_2(u,v) I(i+u, j+v)$$

O_2

Explicații?

Răspunsul filtrului este mare pentru intensități mari în imagine. Dacă am avea o porțiune din imagine numai cu alb acolo am obține cea mai mare valoare (răspunsul filtrului) pentru pixelii din O_2

Găsirea şabloanelor cu filtre

- filtrul f_3 =  normalizat (suma = 0, $f_3 = f_2 - \overline{f_2}$)
- corelație: filtrăm imaginea cu filtrul f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$

I

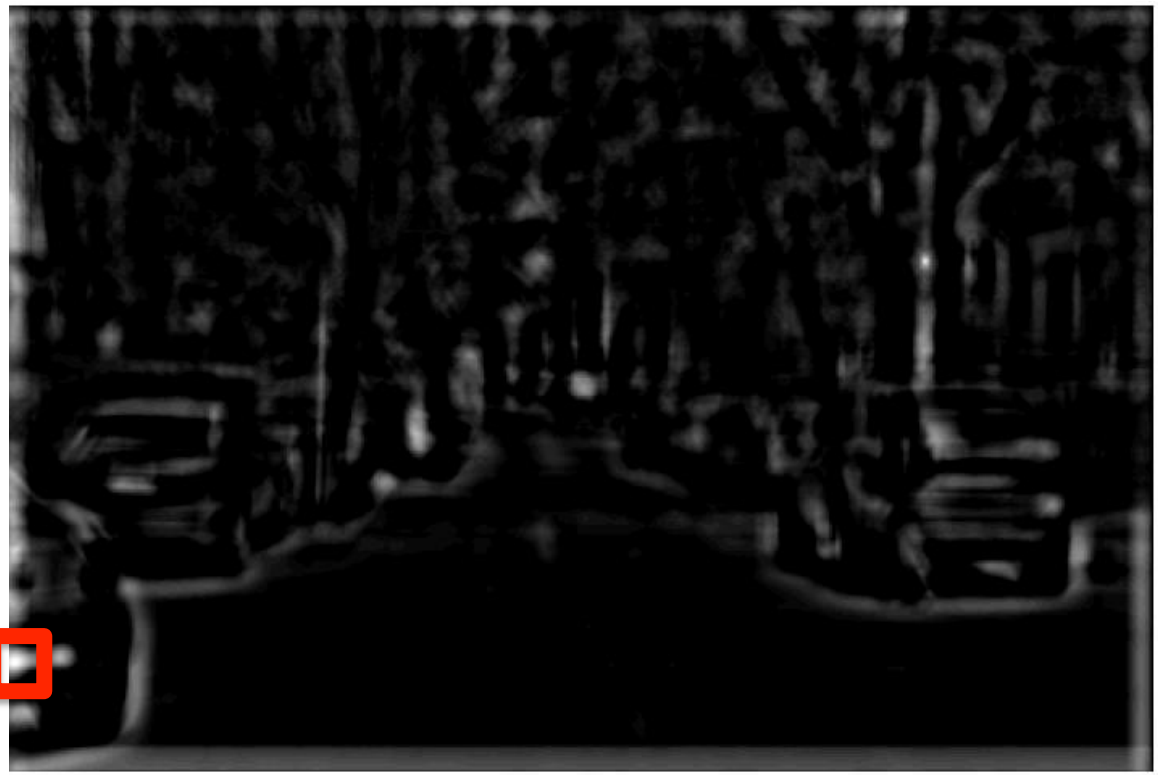
$$O_3 = f_3 \otimes I$$

Ce rezultă după filtrare?

Găsirea şabloanelor cu filtre

- filtrul f_3 =  normalizat (suma = 0, $f_3 = f_2 - \overline{f_2}$)
- corelație: filtrăm imaginea cu filtrul f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$



O_3

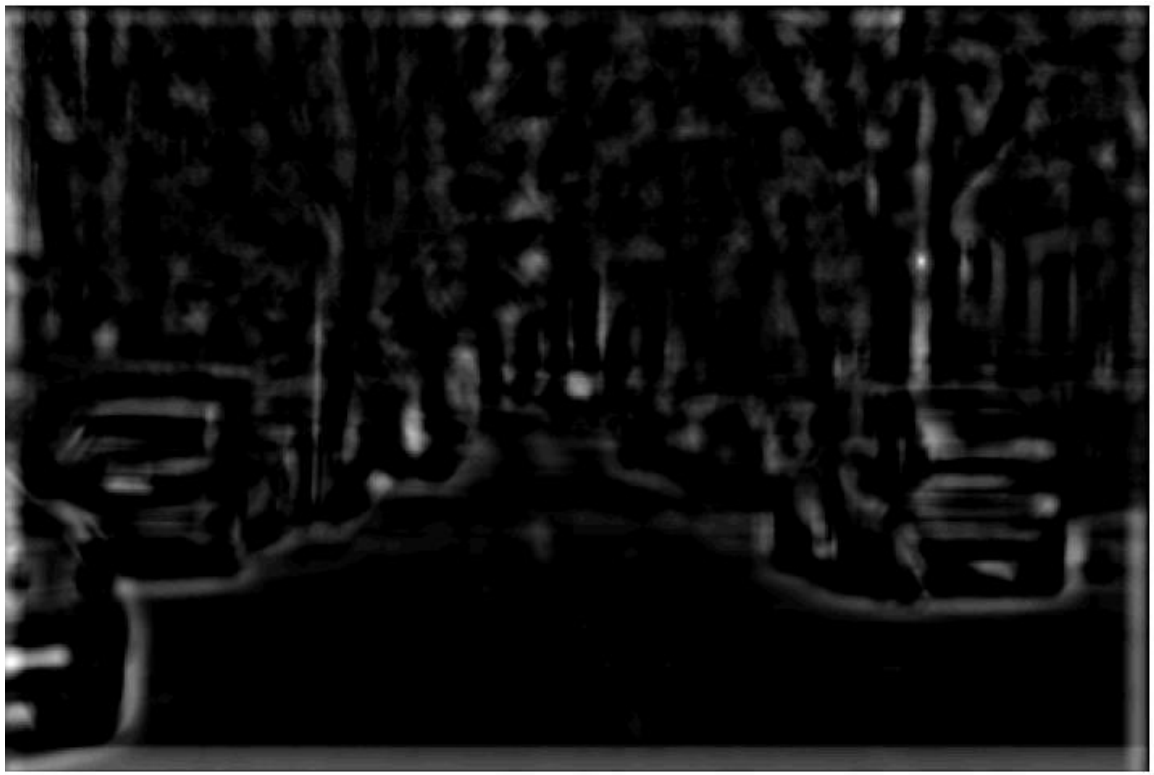
Explicații?

Răspunsul filtrului este mare atunci când valorilor mici (negative) din filtru corespund intensități mici (pixeli negri) din imagine, iar valorilor mari (pozitive) din filtru corespund intensități mari (pixeli albi) din imagine.

Găsirea şabloanelor cu filtre

- filtrul f_3 = ⬆ normalizat (suma = 0, $f_3 = f_2 - \overline{f_2}$)
- corelație: filtrăm imaginea cu filtrul f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$



O_3

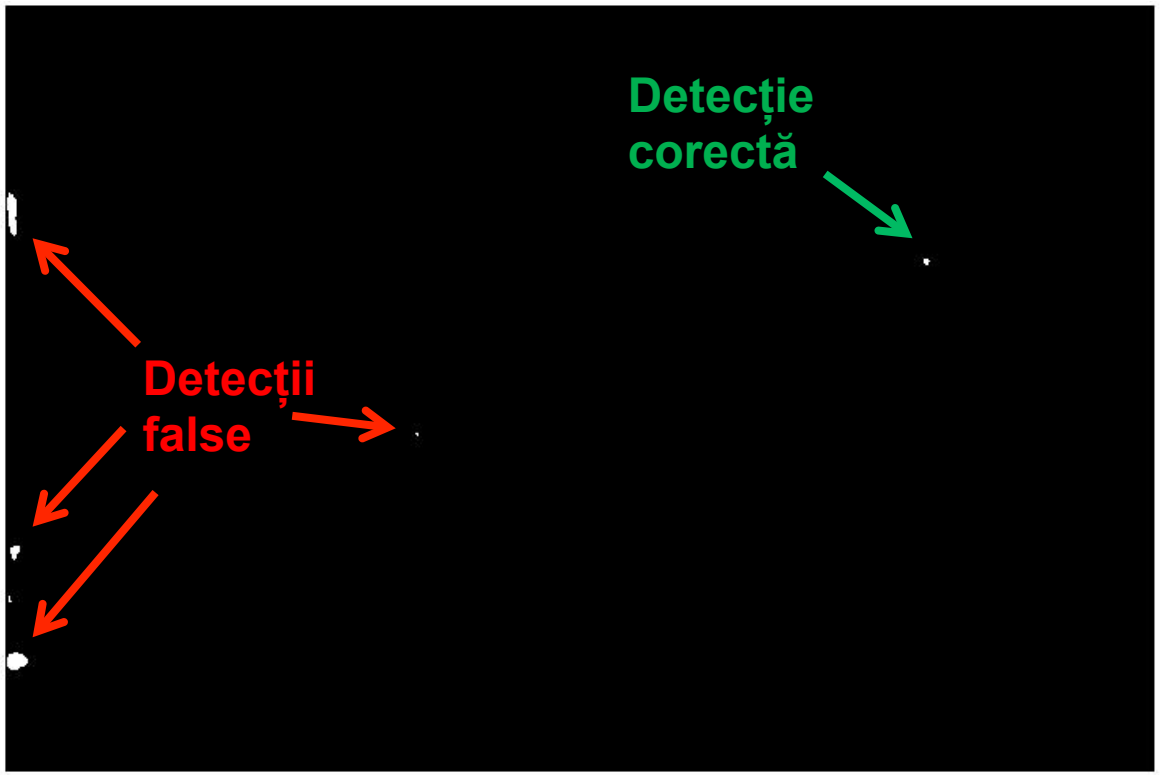
Explicații?

Răspunsul filtrului este maxim atunci când avem în imagine o fereastră de dimensiunile filtrului cu pixeli albi și negri corelați cu semnele +/- ale valorilor filtrului.

Găsirea şabloanelor cu filtre

- filtrul f_3 = ⬆ normalizat (suma = 0, $f_3 = f_2 - \overline{f_2}$)
- corelație: filtrăm imaginea cu filtrul f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$



O_3

Thresholding

Aplicăm un prag (threshold) imaginii filtrate obținute: găsește toți pixelii cu intensitate > threshold

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- suma pătratelor distanțelor

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

I

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- suma pătratelor distanțelor

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

afișează alături
1 – imaginea normalizata
(imparte la maxim)

Găsește valoarea minimă detectând șablonul.

O_4

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- suma pătratelor distanțelor

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

Efect de umbrire al imaginii care afectează șablonul.

Ce obținem?

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- suma pătratelor distanțelor

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

Explicații?

Răspunsul filtrului este senzitiv la intensitatea medie din fereastră

O_5

Găsirea şabloanelor cu filtre

- filtrul f_1 = ⬆
- corelație normalizată

Matlab: O = normxcorr2(f1,I);

media intensităților în ferestra curentă din imagine

media intensităților filtrului

$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$

deviația standard a intensităților în ferestra curentă din imagine

deviația standard a intensităților în filtru

Găsirea şabloanelor cu filtre

- filtru f_1 = 
- corelație normalizată

Matlab: O = normxcorr2(f1,I);

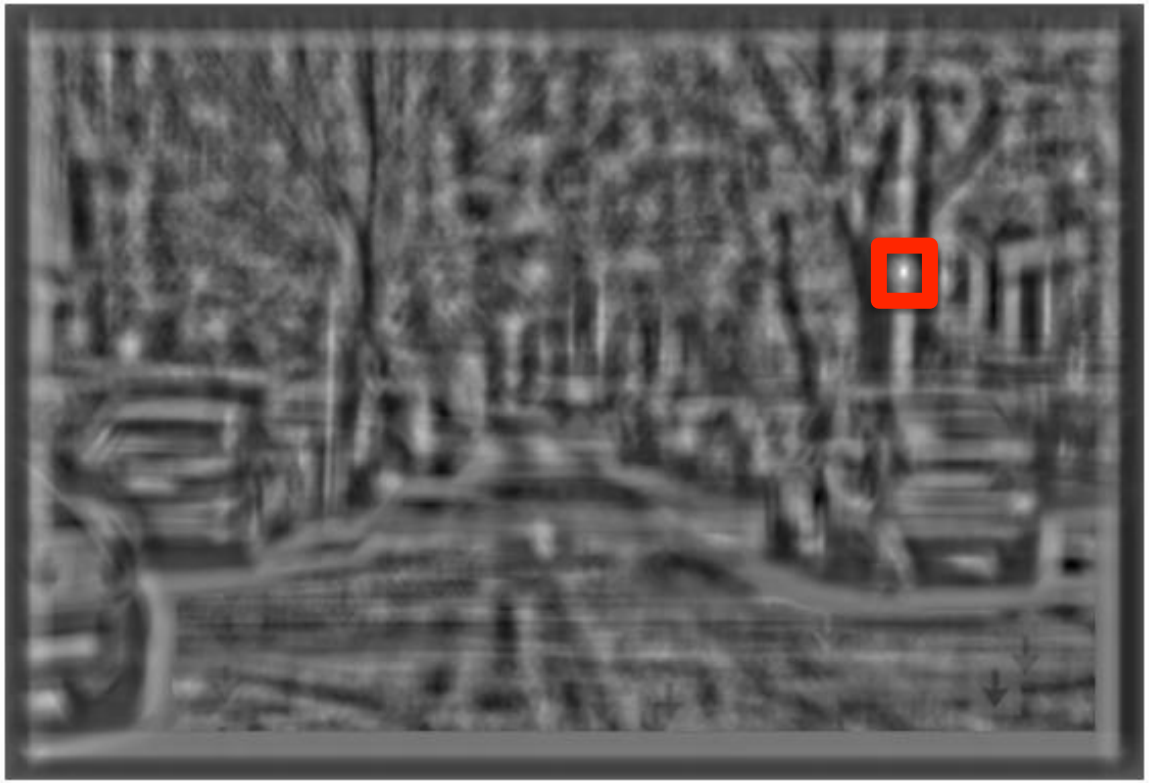
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$

Găsirea şabloanelor cu filtre

- filtru f_1 = 
- corelație normalizată

Matlab: O = normxcorr2(f1,I);

$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$



Găsește valoarea minimă detectând șablonul.

Găsirea şabloanelor cu filtre

- filtru f_1 = 
- corelație normalizată

Matlab: O = normxcorr2(f1,I);

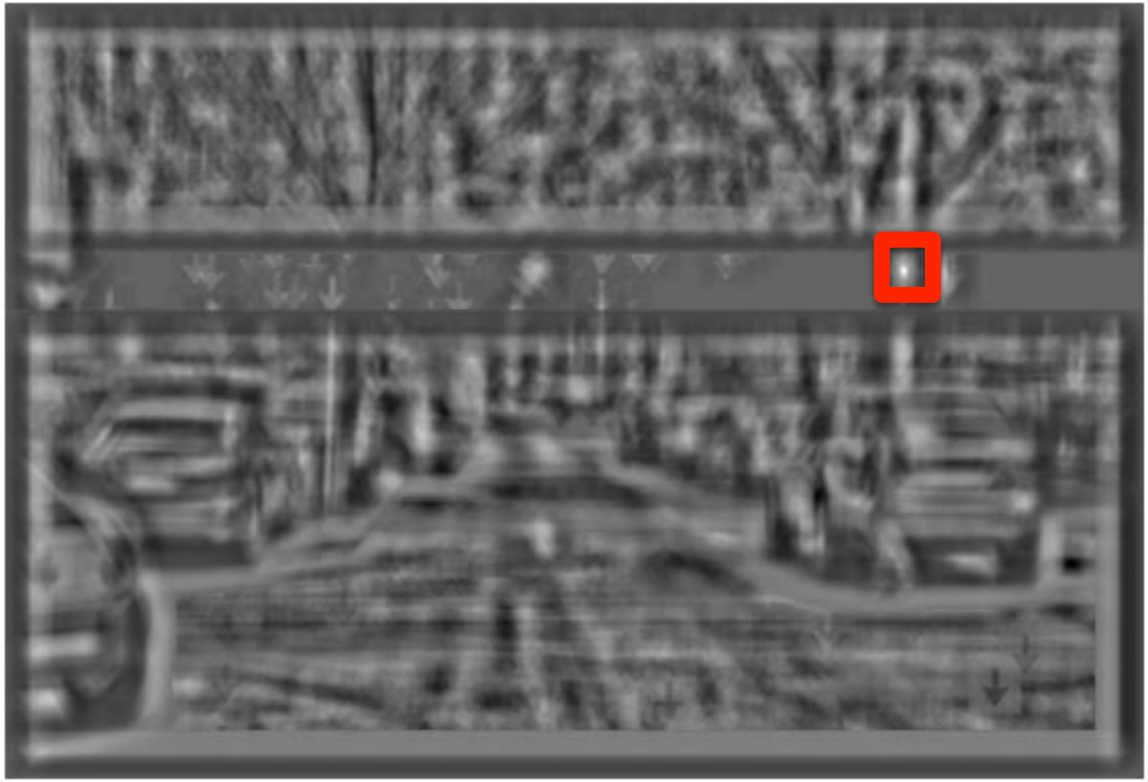
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$

Găsirea şabloanelor cu filtre

- filtru f_1 = 
- corelație normalizată

Matlab: O = normxcorr2(f1,I);

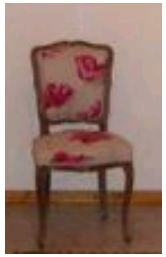
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$



Găsește valoarea minimă detectând șablonul.

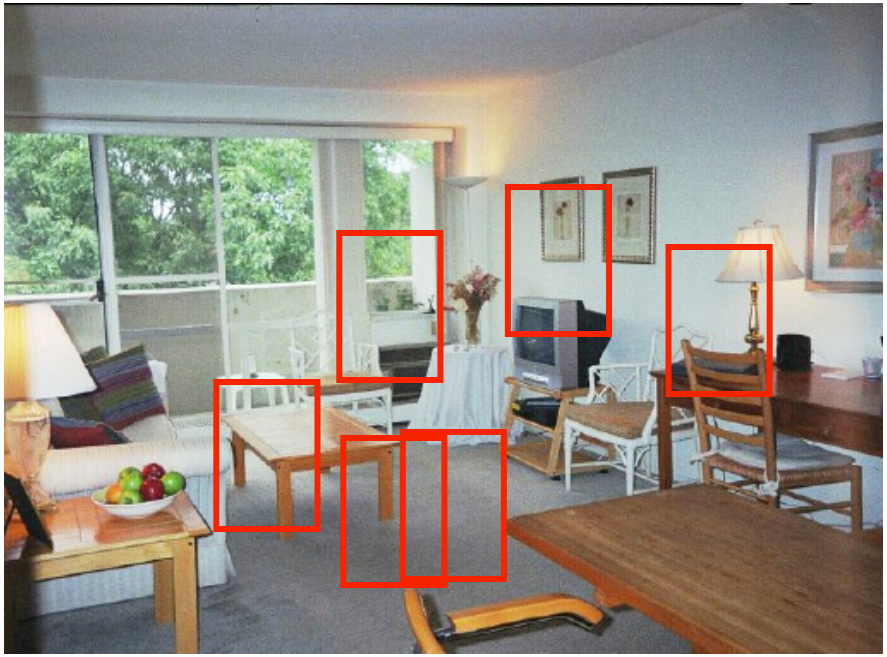
Cea mai bună măsură a similarității?

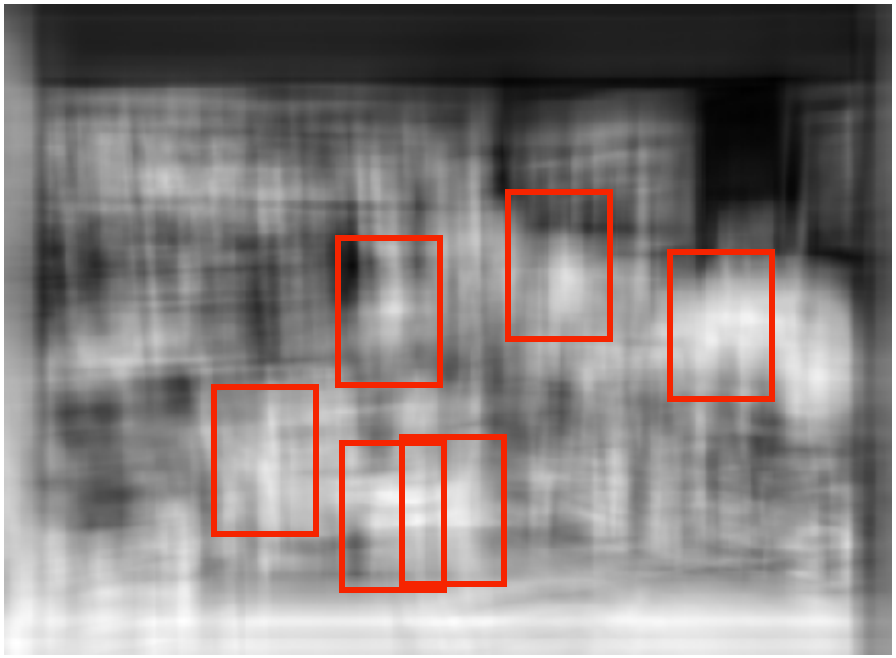
- filtru de medie 0: rezultate nu prea bune(apar detecții false)
- suma pătratelor distanțelor: senzitiv la intensitatea medie
- corelație normalizată: invariant la intensitatea medie şi la contrast



Detectare de obiecte cu găsirea șabloanelor

Găsiți scaunele în această imagine





Găsirea şabloanelor nu rezolvă problema

A “popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts.” Nevatia & Binford, 1977.

Detectare de obiecte cu găsirea șabloanelor

Avantaje

- metodă simplă, ușor de implementat
- rezultate bune când șablonul/ceva foarte asemănător cu șablonul se află în imagine

Dezavantaje

- invariant la mărime, rotație (chiar a aceluiași șablon): dacă mărim șablonul sau îl rotim nu mai obținem același rezultat
- pentru clase de obiecte cu variabilitate în înfățișare foarte mare (mașini, persoane) metoda nu e aplicabilă
- nu putem să folosim această metodă la detectarea facială

Dacă vrem să găsim un şablon mai mic sau mai mare?

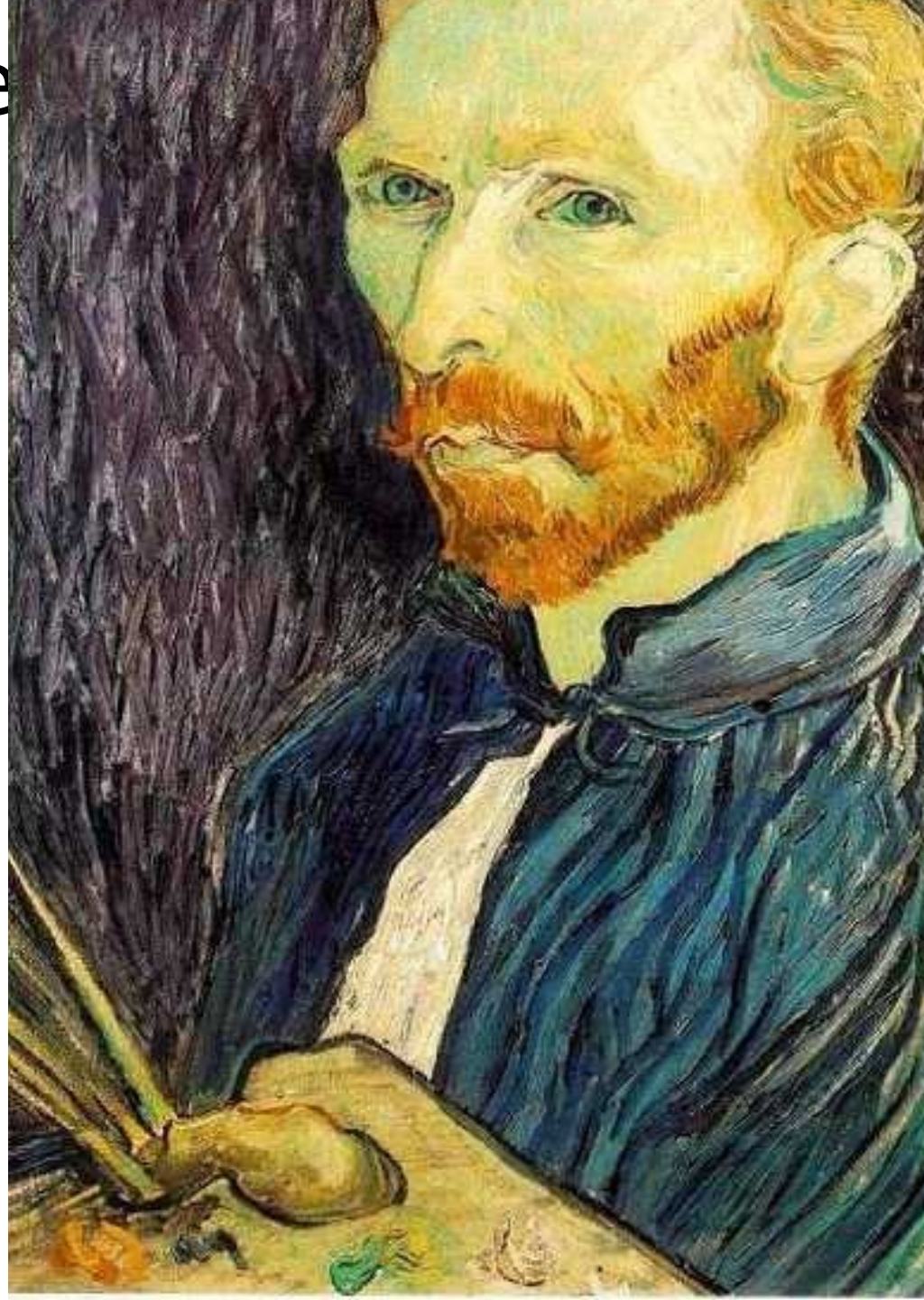
şablon inițial 

şablon mai mic  alternativă → construim o piramidă de imagini redimensionate

Redimensionarea imaginilor

Micşorare

Această imagine este prea mare pentru a încăpea pe slide. Cum putem genera o imagine de 2x mai mică?



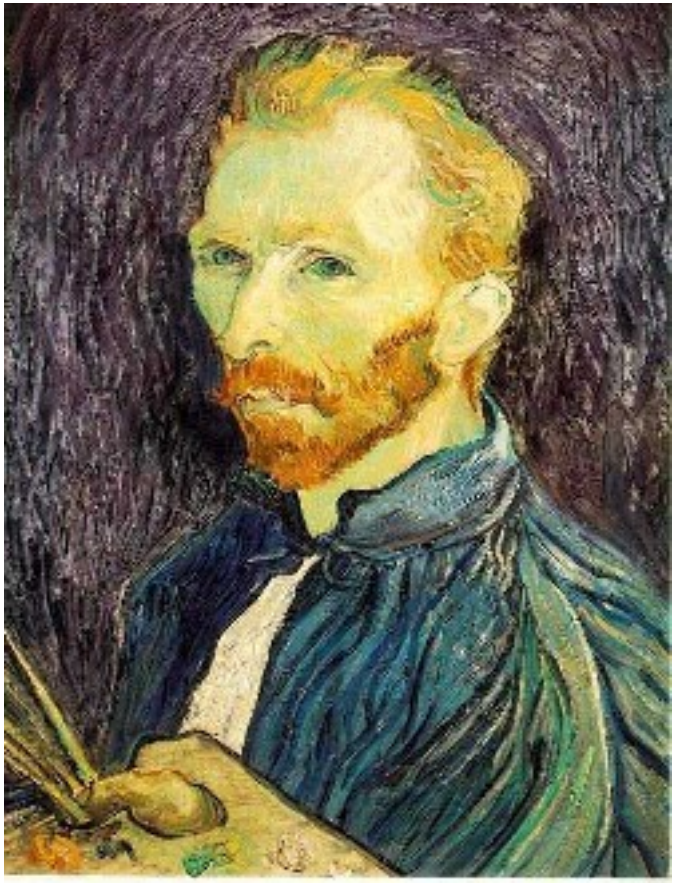
Slide adaptat după Steve Seitz

Algoritm pentru micşorarea imaginilor cu factor 2

1. Input: imagine de dimensiuni L x C
2. Selectează fiecare al doilea pixel

```
imgRedusa = img (1:2:end, 1:2:end);
```

Rezultate



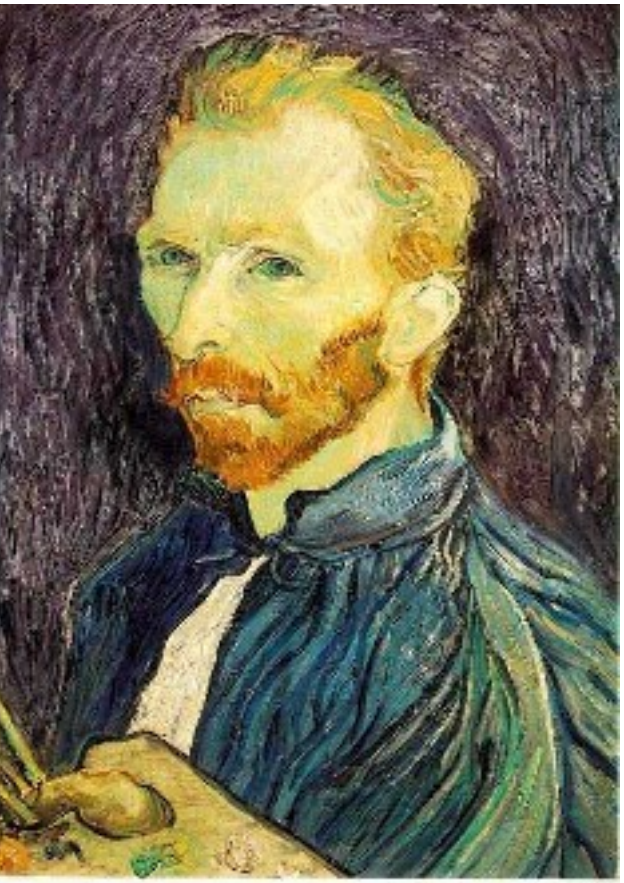
Eliminăm fiecare a doua linie şi a doua coloană pentru a crea o imagine de 2x mai mică

1/4

1/8

Slide adaptat după Steve Seitz

Rezultate - zoom



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Observații?

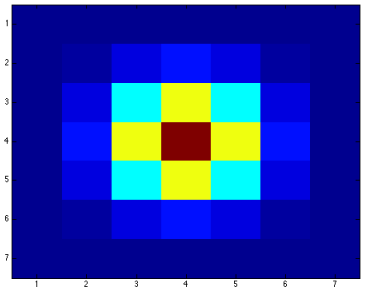
Slide adaptat după Steve Seitz

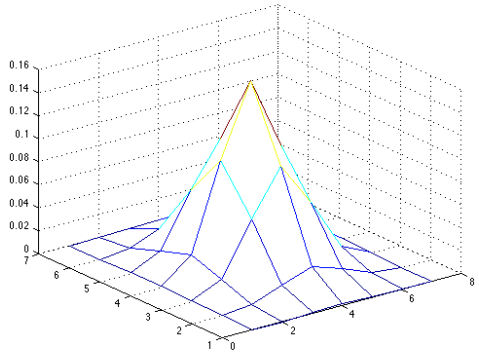
Algoritm pentru micşorarea imaginilor cu factor 2 cu filtrare Gaussiană

1. Input: imagine de dimensiuni L x C
2. Filtrează imaginea cu un filtru Gaussian

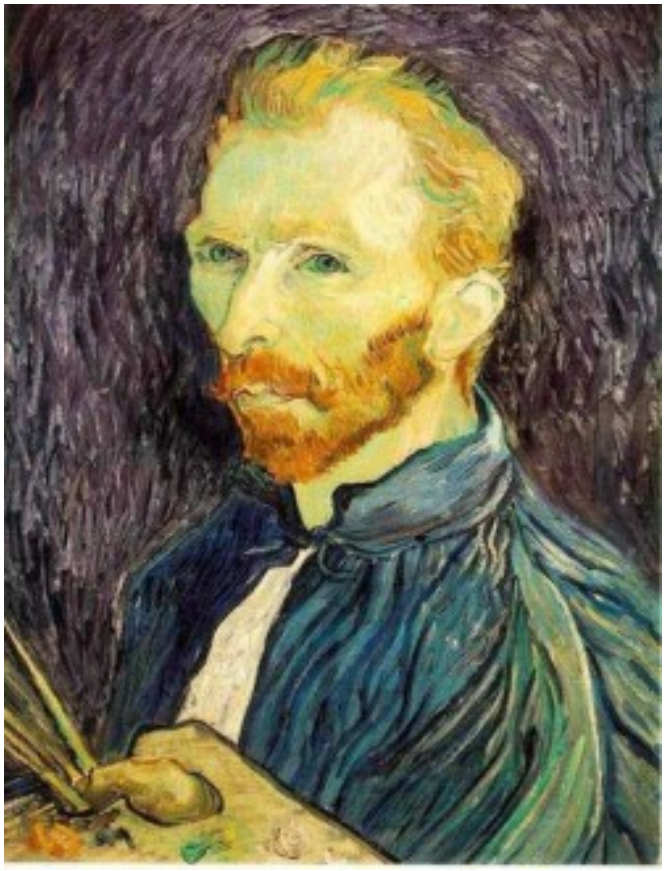
   imgBlurata = imfilter(img, fspecial('gaussian',7,1));
3. Selectează fiecare al doilea pixel

   imgRedusa = imgBlurata (1:2:end, 1:2:end);





Rezultate



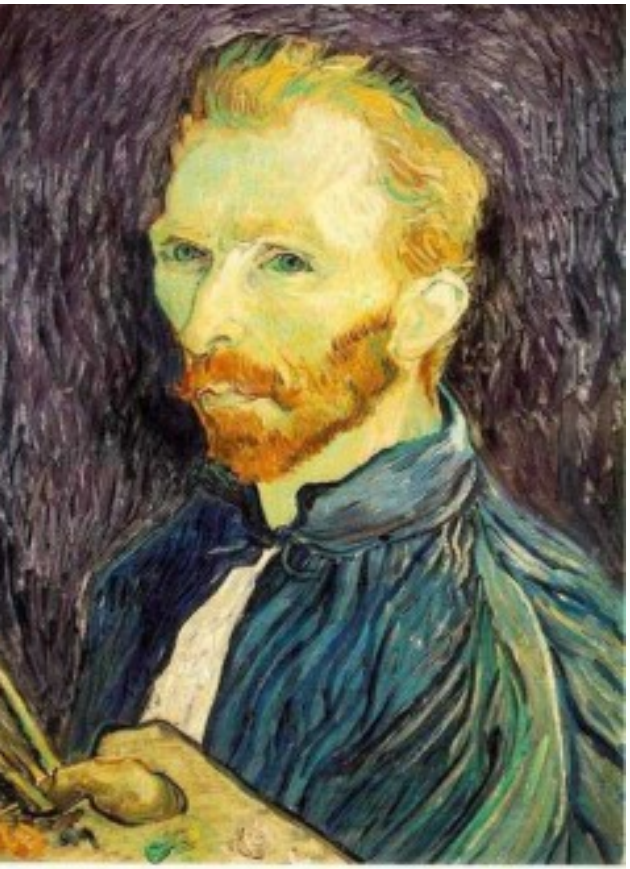
Gaussian 1/2

G 1/4

G 1/8

Slide adaptat după Steve Seitz

Rezultate - zoom



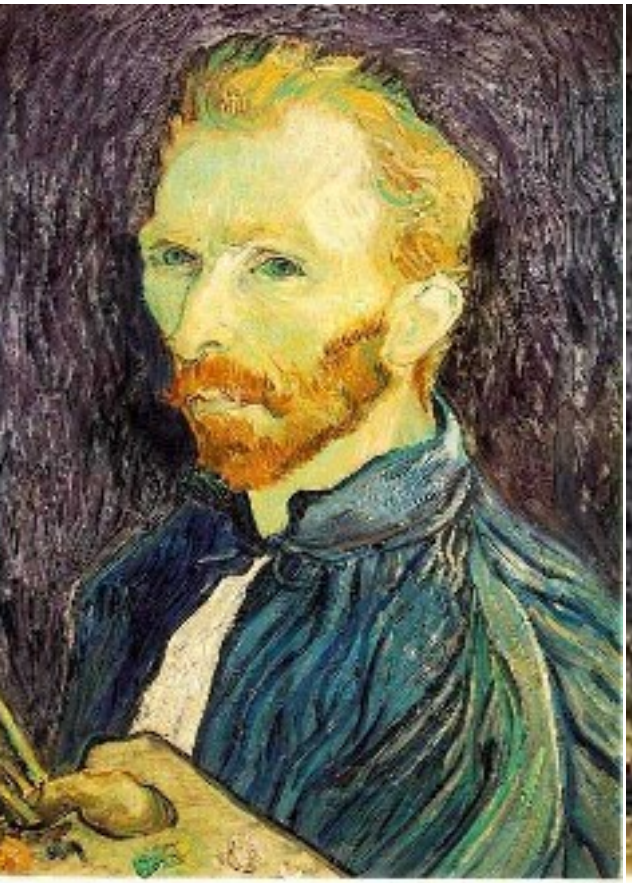
Gaussian 1/2

G 1/4

G 1/8

Slide adaptat după Steve Seitz

Versus...



1/2

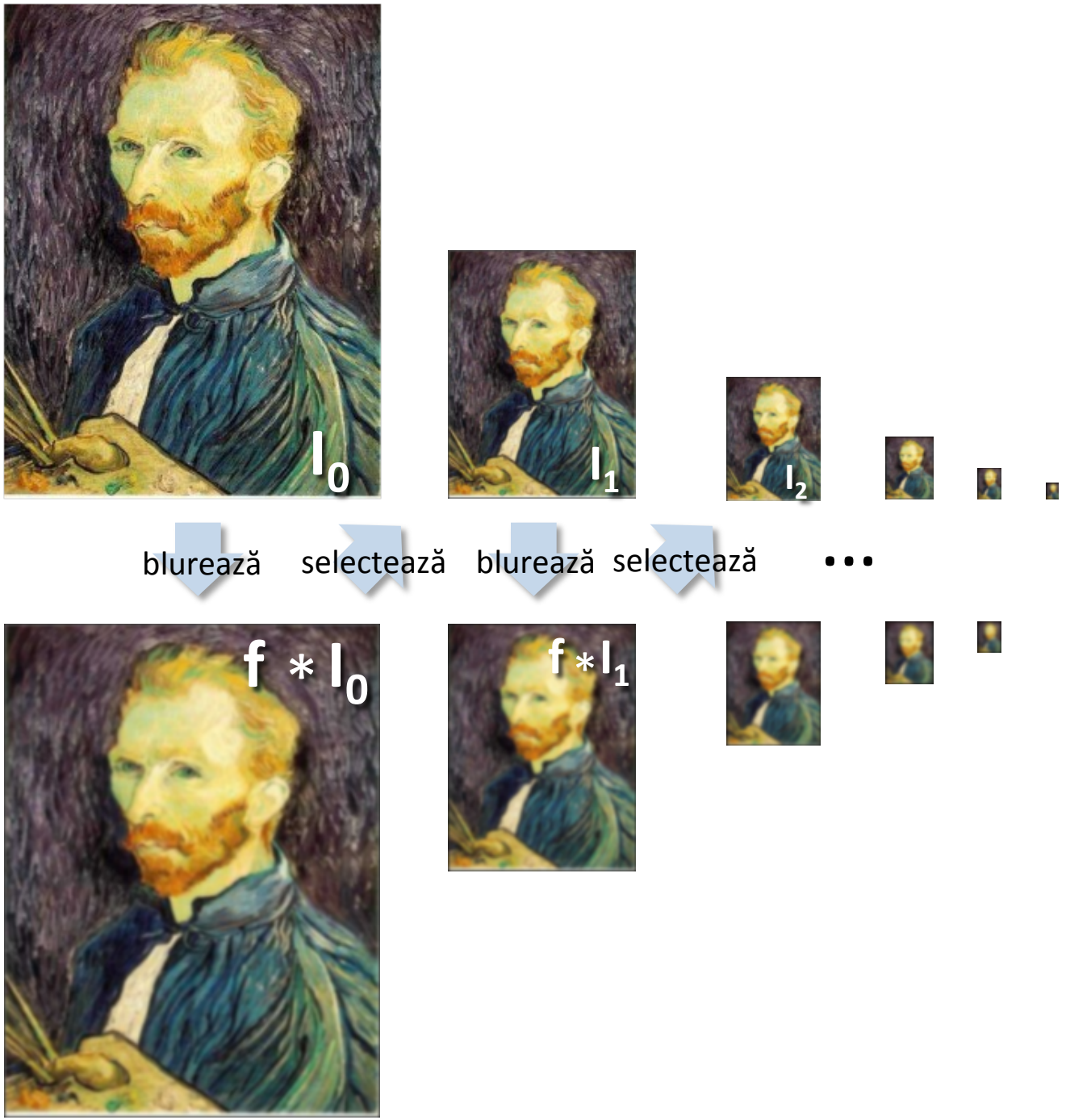
1/4 (2x zoom)

1/8 (4x zoom)

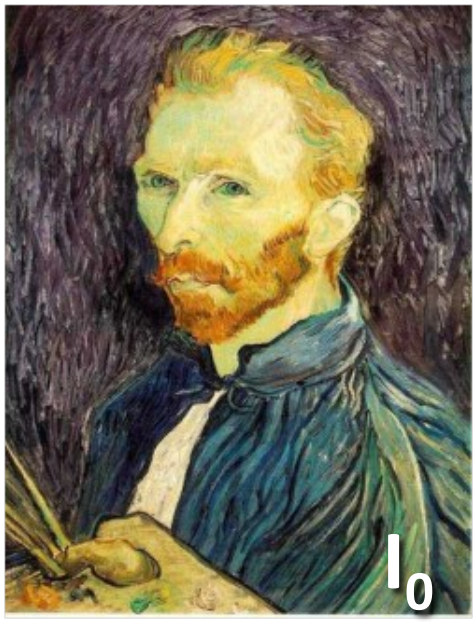
Slide adaptat după Steve Seitz

Filtrare Gaussiană

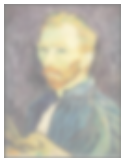
- Soluție: filtrează mai întâi imaginea cu un filtru Gaussian, *apoi* selectează pixelii din 2 în 2



Piramidă Gaussiană



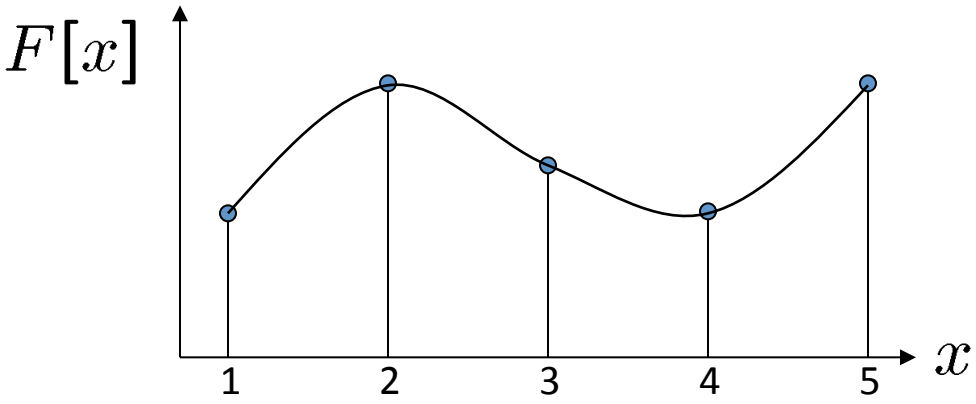
blur selectează blur selectează • • •

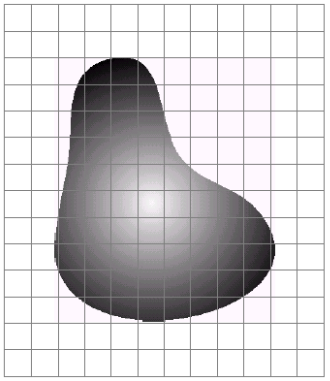


Mărirea imaginilor

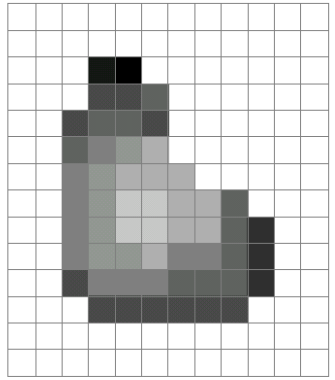
- imagine prea mică pentru ecran: 
  (45 x 45 pixeli)
- cum putem să o mărim de 10 ori?
  (450 x 450 pixeli)
- metodă simplă: repetăm fiecare rând şi coloană de 10 ori (fiecare pixel multiplicat de 100 de ori)
- interpolare “cel mai apropiat vecin”

Interpolarea imaginilor





f - funcție continuă



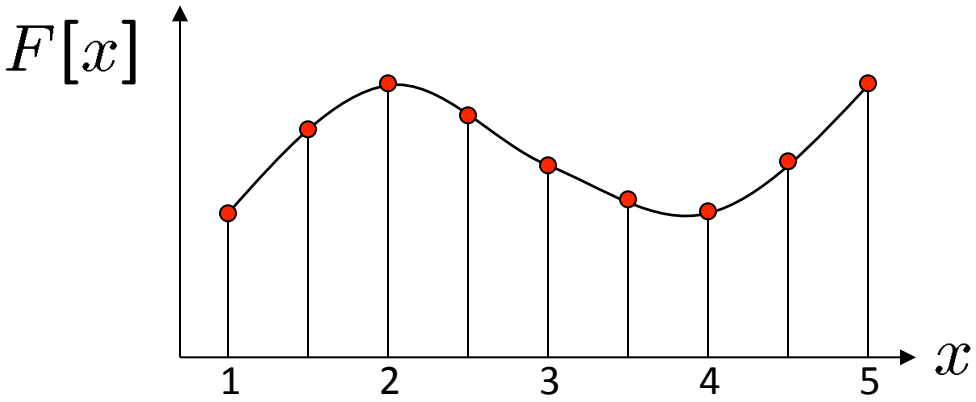
F - funcție discretă

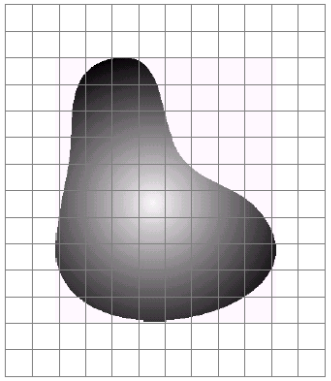
Cum se formează imaginile digitale?

- eșantionare : discretizăm spațiul în pixeli
- trecem de la o funcție continuă *f* la o funcție discretă F
- dacă am putea reconstrui funcția continuă inițială *f* , am putea genera imagini la orice rezoluție

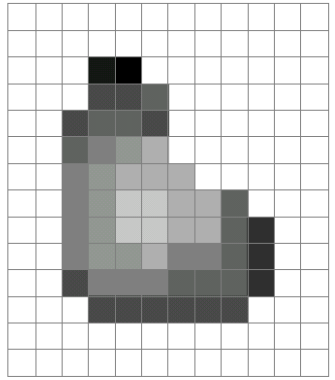
Slide adaptat după Steve Seitz

Interpolarea imaginilor





f - funcție continuă



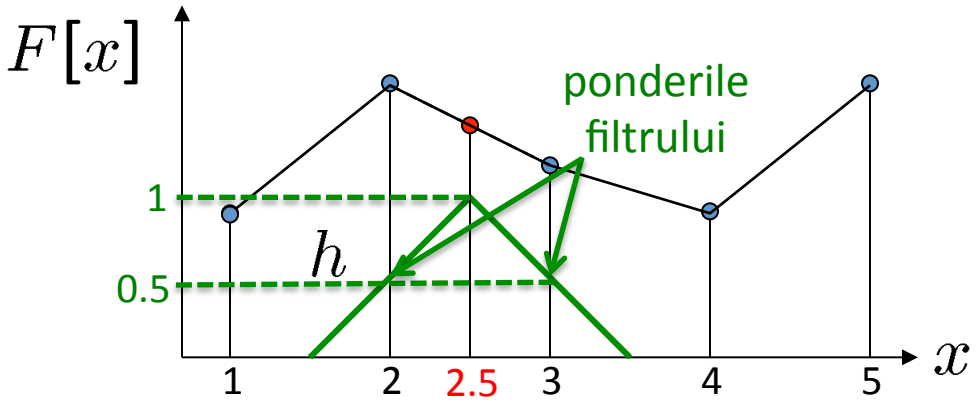
F - funcție discretă

Cum se formează imaginile digitale?

- eșantionare : discretizăm spațiul în pixeli
- trecem de la o funcție continuă *f* la o funcție discretă F
- dacă am putea reconstrui funcția continuă inițială *f* , am putea genera imagini la orice rezoluție

Slide adaptat după Steve Seitz

Interpolarea imaginilor



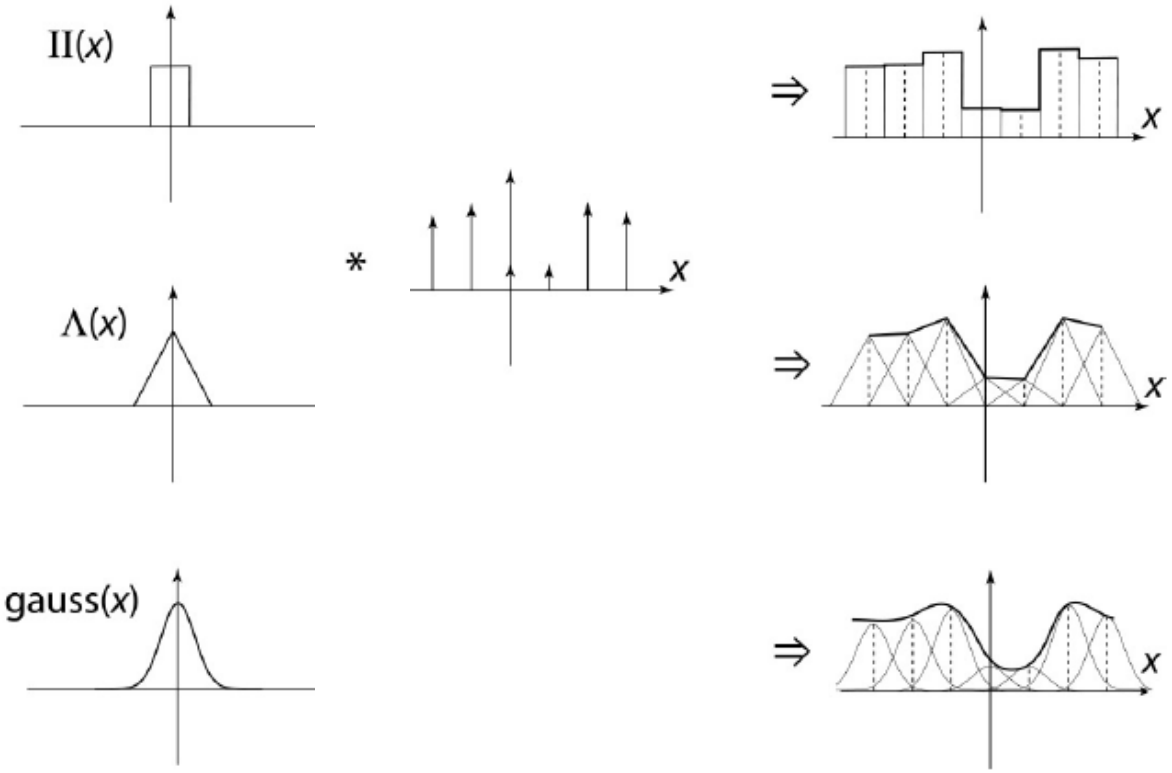
h – filtru liniar

- Dacă nu cunoaştem *f* ?
  - aproximăm f: $\tilde{f}$
  - folosim filtrarea
  - convertim F în funcția f_F = { F(x), dacă x e întreg; 0, altfel }
  - reconstruim f prin convoluție cu un filtru de reconstrucție h

$$\tilde{f} = h * f_F$$

Slide adaptat după Steve Seitz

Tipuri de interpolare



Cel mai apropiat vecin

Liniară

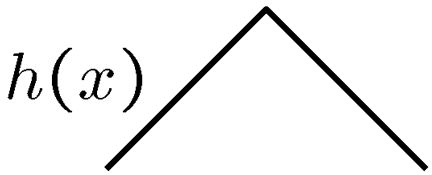
Gaussiană

Slide adaptat după B. Curless

Filtre de reconstrucție

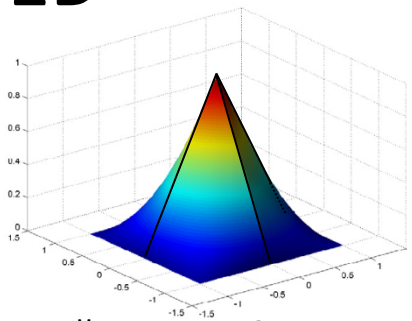
- 1D → 2D

$h(x)$



realizează **interpolare liniară**
pe baza celor mai apropiați 2 pixeli

$h(x,y)$



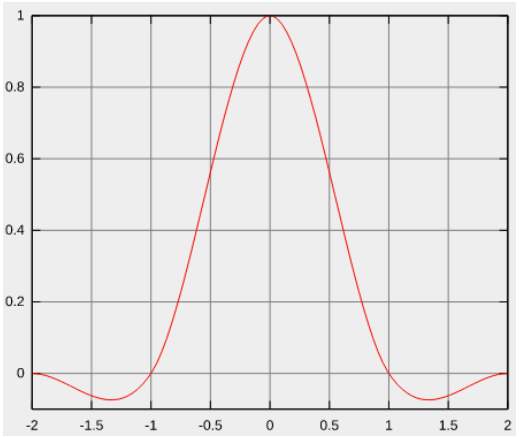
realizează **interpolarea biliniară** pe baza celor mai apropiați 4 pixeli

$$x_1 = \lfloor x \rfloor \qquad f_{11} \equiv f(x_1, y_1)$$
$$x_2 = \lfloor x \rfloor + 1 \qquad f_{12} \equiv f(x_1, y_2)$$
$$y_1 = \lfloor y \rfloor \qquad f_{21} \equiv f(x_2, y_1)$$
$$y_2 = \lfloor y \rfloor + 1 \qquad f_{22} \equiv f(x_2, y_2)$$

$$f_{y1} = f_{11} + \frac{f_{21} - f_{11}}{x_2 - x_1}(x - x_1)$$

$$f_{y2} = f_{12} + \frac{f_{22} - f_{12}}{x_2 - x_1}(x - x_1)$$

$$f(x,y) = f_{y1} + \frac{f_{y2} - f_{y1}}{y_2 - y_1}(y - y_1)$$



$$f(x; -1 \le a < 0) = \begin{cases} (a+2)|x|^3 - (a+3)x^2 + 1 & \text{for } 0 \le |x| \le 1 \\ a|x|^3 - 5ax^2 + 8a|x| - 4a & \text{for } 1 < |x| \le 2 \\ 0 & \text{otherwise} \end{cases}$$

realizează **interpolare cubică**
pe baza celor mai apropiați 4 pixeli

2D →

realizează interpolare bicubică
pe baza celor mai apropiați 16 pixeli

Interpolarea imaginilor

Imagine inițială:  x 10

Interpolare “cel mai apropiat vecin”

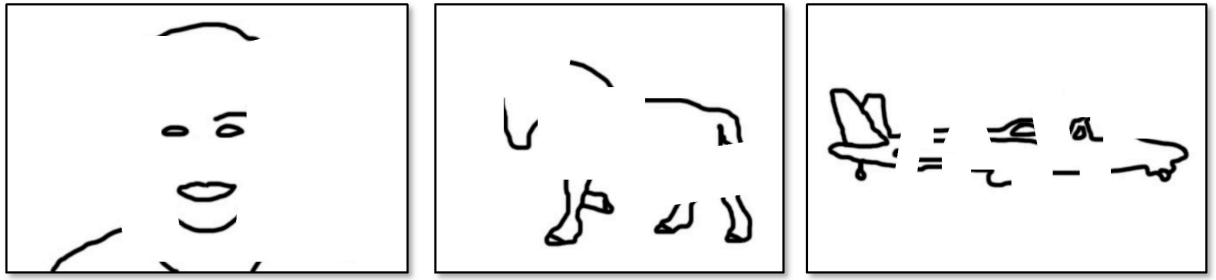
Interpolare biliniară

Interpolare bicubică

Gradienți & muchii

Detectarea muchiilor

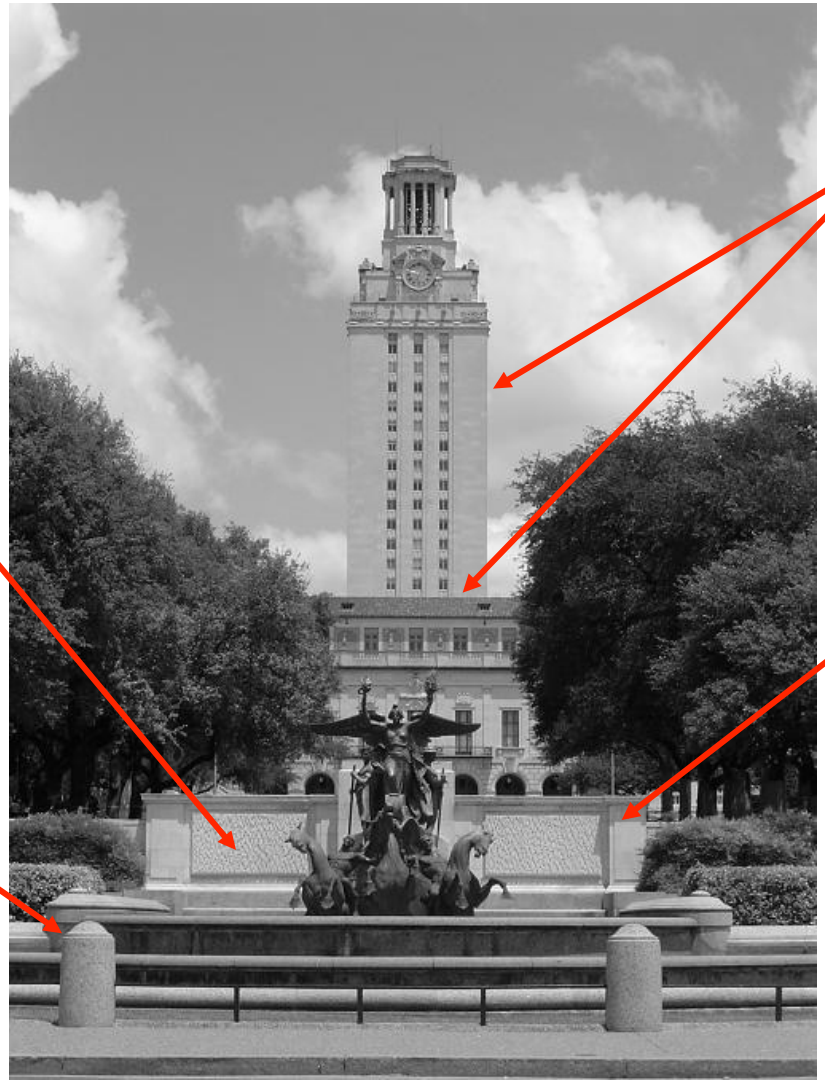
- **Scop**: transformăm imaginea dintr-o matrice de pixeli într-o mulțime de curbe/segmente/contururi
- **De ce?**



- **Idee principală**: localizare gradienți, post-procesare

Slide adaptat după K. Grauman

Cum se formează muchiile?



Discontinuități în 3D: conturul obiectelor

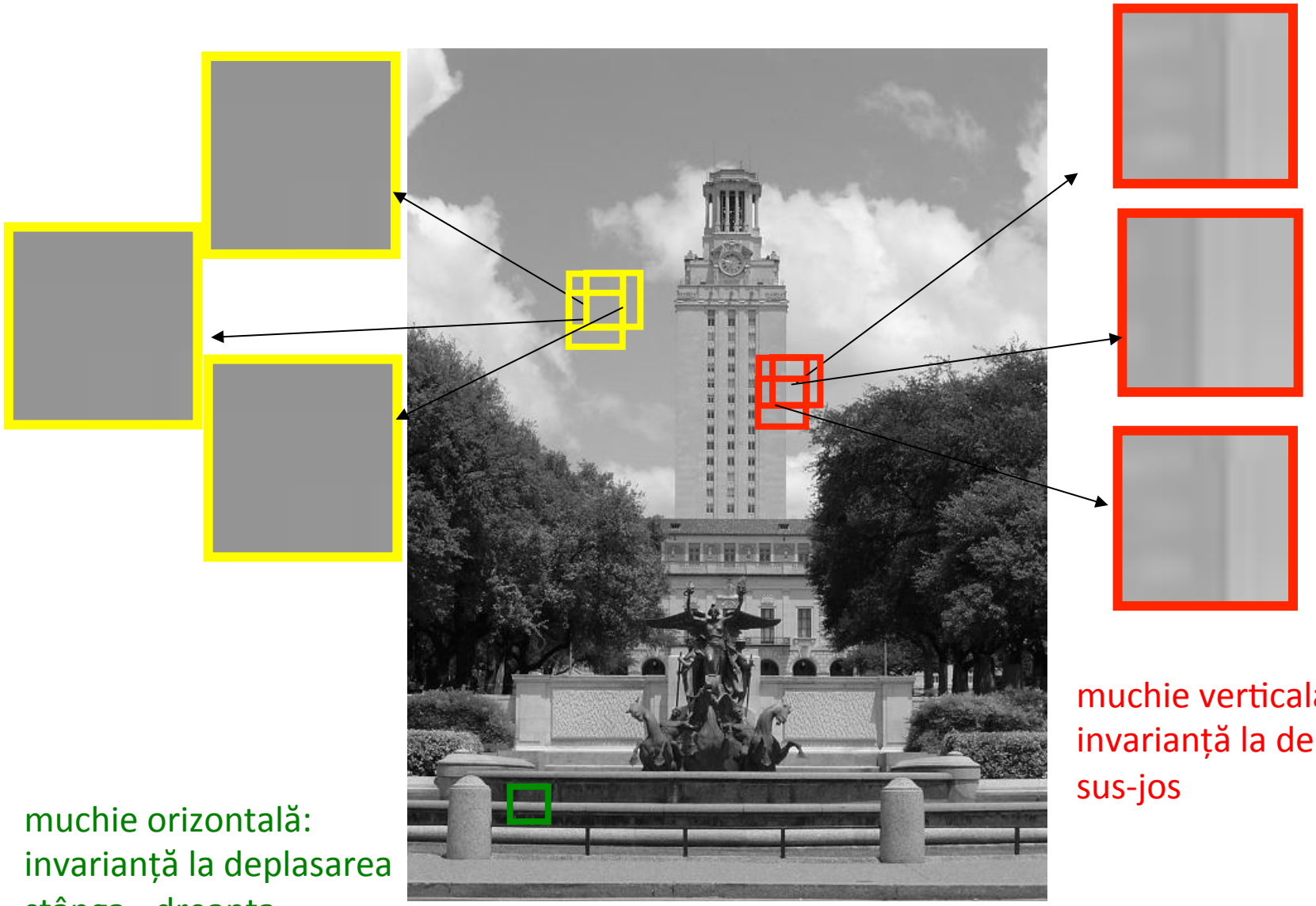
Reflexia luminii în funcție de textură

Formarea umbrelor

Discontinuitatea în orientarea suprafeței

Slide adaptat după K. Grauman

Muchii/gradienți şi invarianță



muchie verticală: invarianță la deplasarea sus-jos

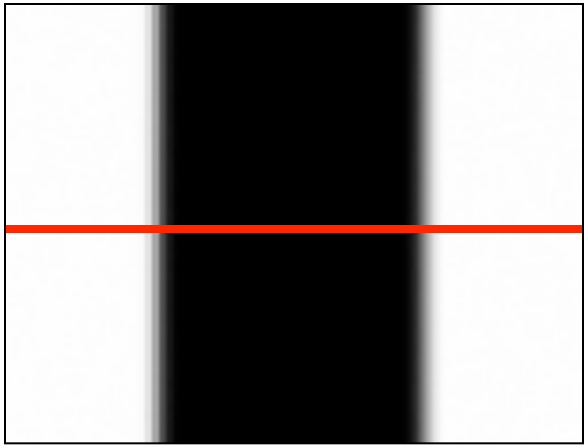
muchie orizontală: invarianță la deplasarea stânga - dreapta

Slide adaptat după K. Grauman

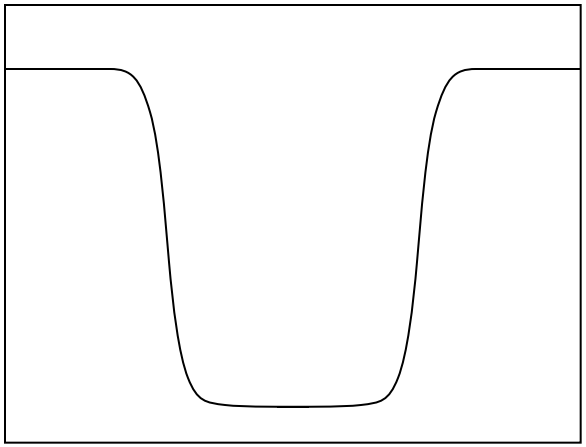
Derivate şi muchii

O muchie este locul în care se produce o schimbare bruscă a funcției de intensitate

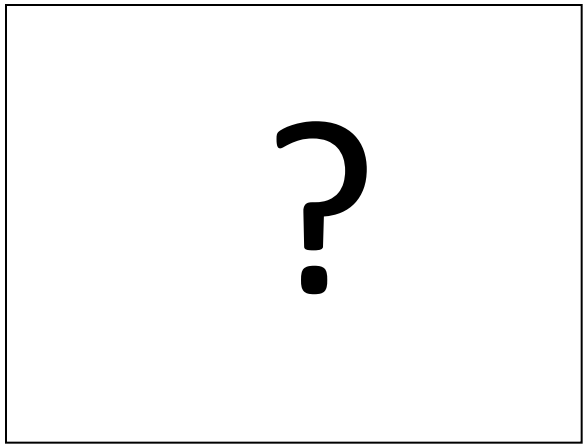
imagine



funcția de intensitate
(de-a lungul liniei roşii)



derivata funcției
de intensitate

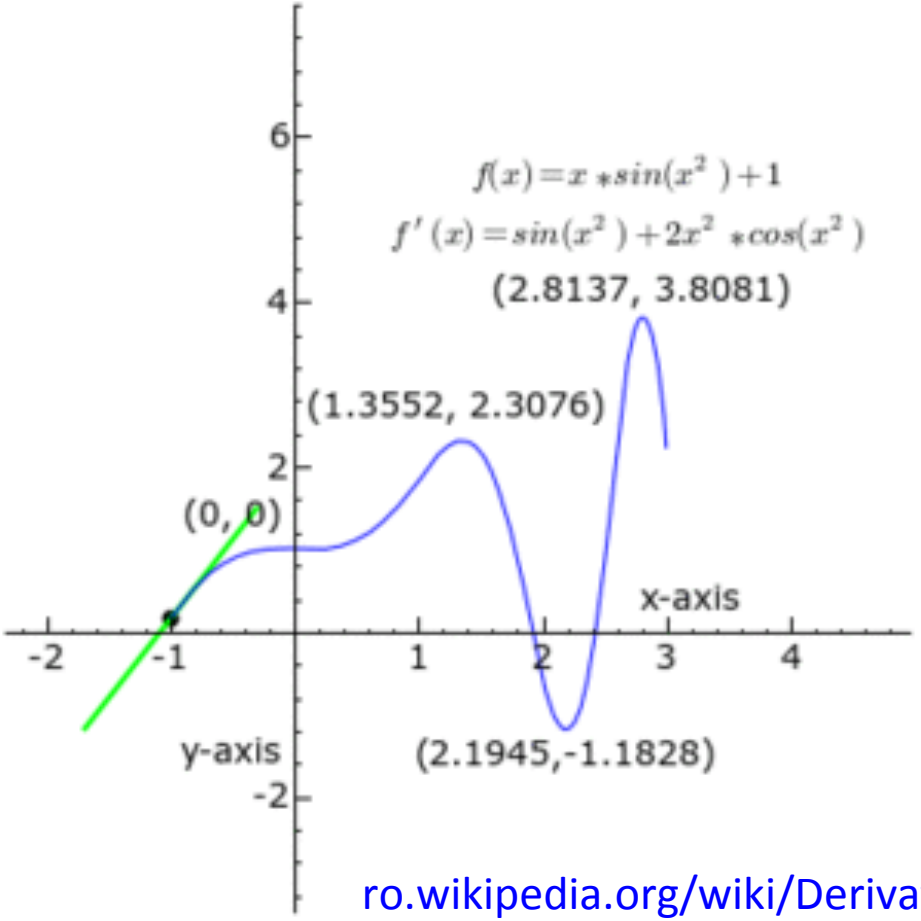


Slide adaptat după S. Lazebnik

Derivata unei funcții

- panta (înclinarea) dreptei tangente la grafic

- **tangenta verde > 0**
- **tangenta neagră = 0**
- **tangenta roşie < 0**

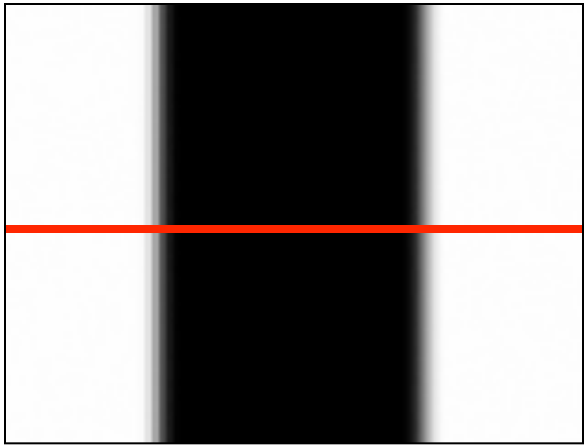


ro.wikipedia.org/wiki/Derivată

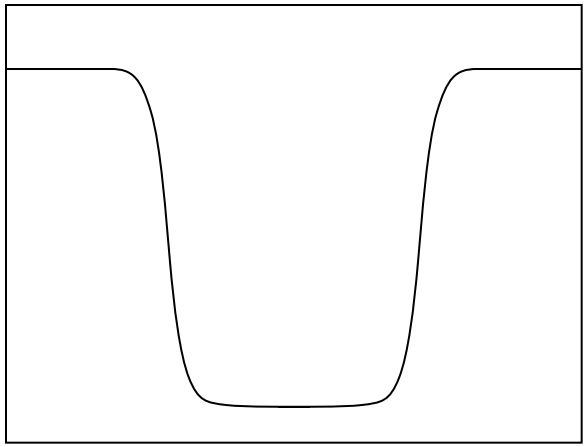
Derivate şi muchii

O muchie este locul în care se produce o schimbare bruscă a funcției de intensitate

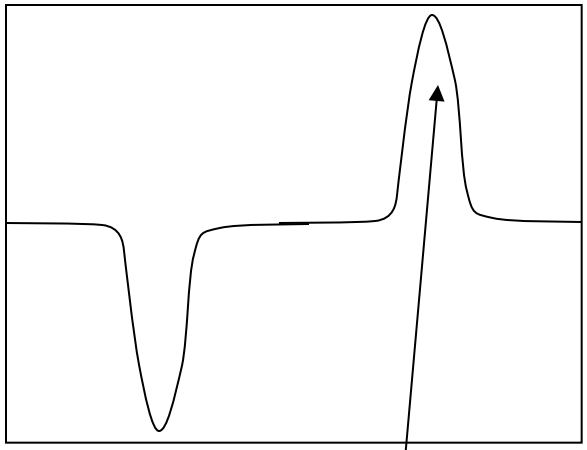
imagine



funcția de intensitate
(de-a lungul liniei roşii)



derivata funcției
de intensitate



muchiile corespund punctelor
de extrem ale derivatei

Slide adaptat după S. Lazebnik

Calculul derivatelor prin corelație/convoluție

Pentru o funcție f(x,y), derivata parțială în raport cu x este:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

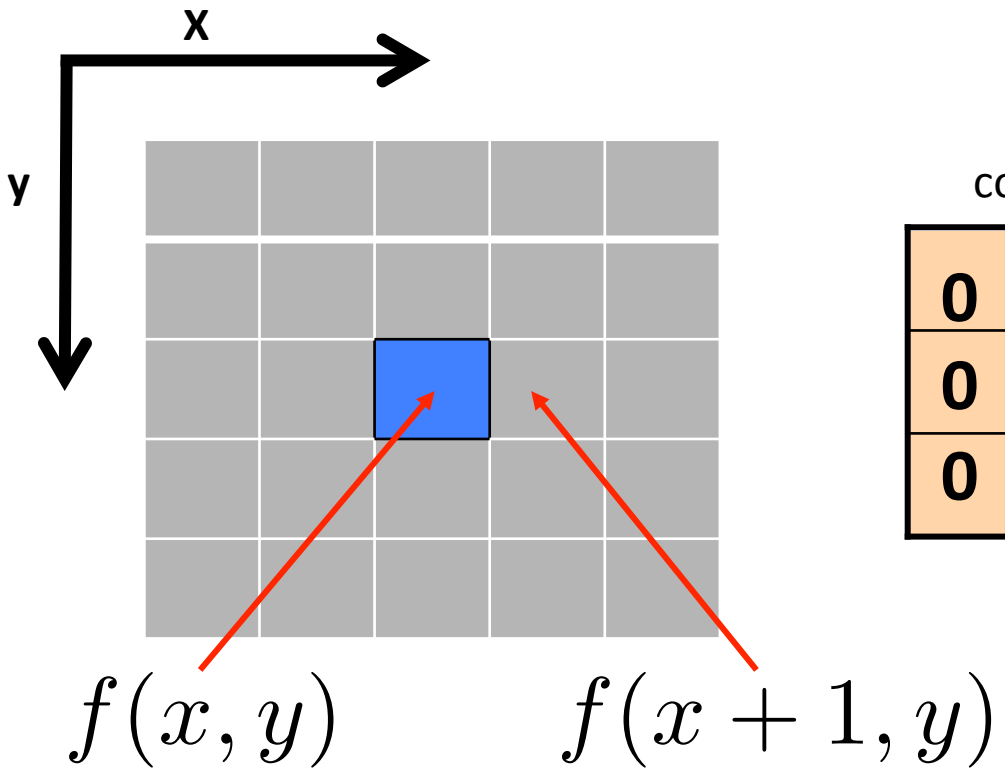
Pentru cazul discret (cazul imaginilor), putem aproxima derivata folosind diferențe finite:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

Care este filtrul de corelație/convoluție care implementează relația de mai sus?

Calculul derivatelor prin corelație/convoluție

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$



corelație

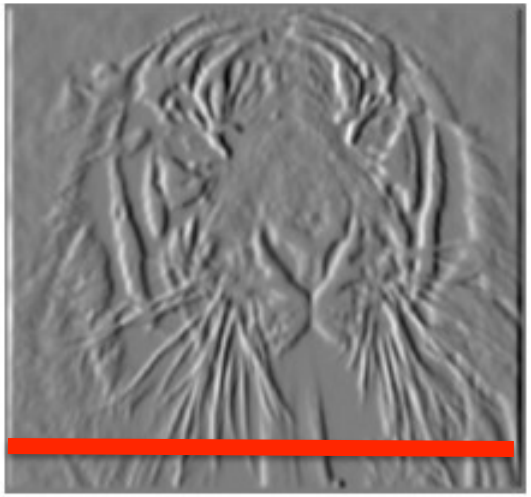
0	0	0
0	-1	1
0	0	0

convoluție

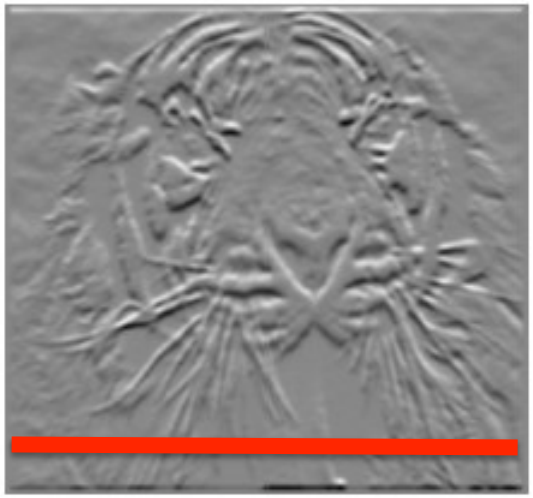
1	-1	

Derivatele parțiale ale unei imagini

$$\frac{\partial f(x,y)}{\partial x}$$



-1	1



$$\frac{\partial f(x,y)}{\partial y}$$

?

-1
1

sau

1
-1

Care imagine arată schimbările în raport cu x/y ?

(arătăm filtrele pentru corelație)

Slide adaptat după S. Lazebnik