Concepte și aplicații în Vederea Artificială

Bogdan Alexe
bogdan.alexe@fmi.unibuc.ro

Radu Ionescu
radu.ionescu@fmi.unibuc.ro

Curs opțional

anul III, semestrul I, 2023-2024

Cursul trecut

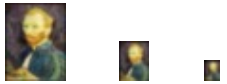
- Diverse modele pentru zgomot în imagini
  - salt and pepper, impuls
  - Gaussian (normal)
- Filtrarea liniară
  - corelație, convoluție
  - filtre: de medie, Gaussian, accentuare
  - aplicație: imagini hibrid

normal

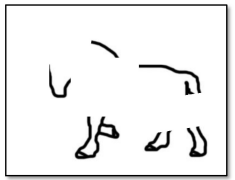
Cursul de azi

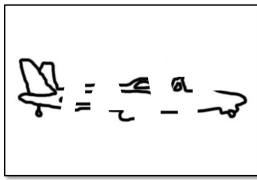
- Filtrarea liniară
  - corelație, convoluție, accentuare
- Filtrarea neliniară
  - filtrul median
- Aplicații ale filtrelor:
  - găsirea şabloanelor (cursul de azi)

  - redimensionarea imaginilor (cursul de azi)



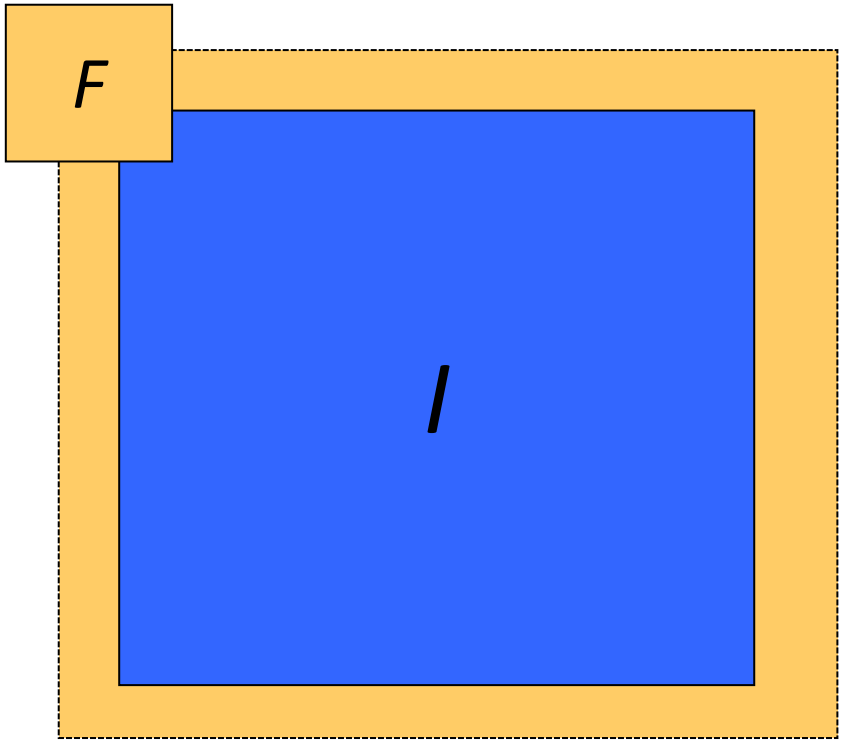
  - extragerea informației (muchii, textură – cursul de azi + săptămâna viitoare)





Corelație

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



Corelație

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

Corelație: $O = F \otimes I$

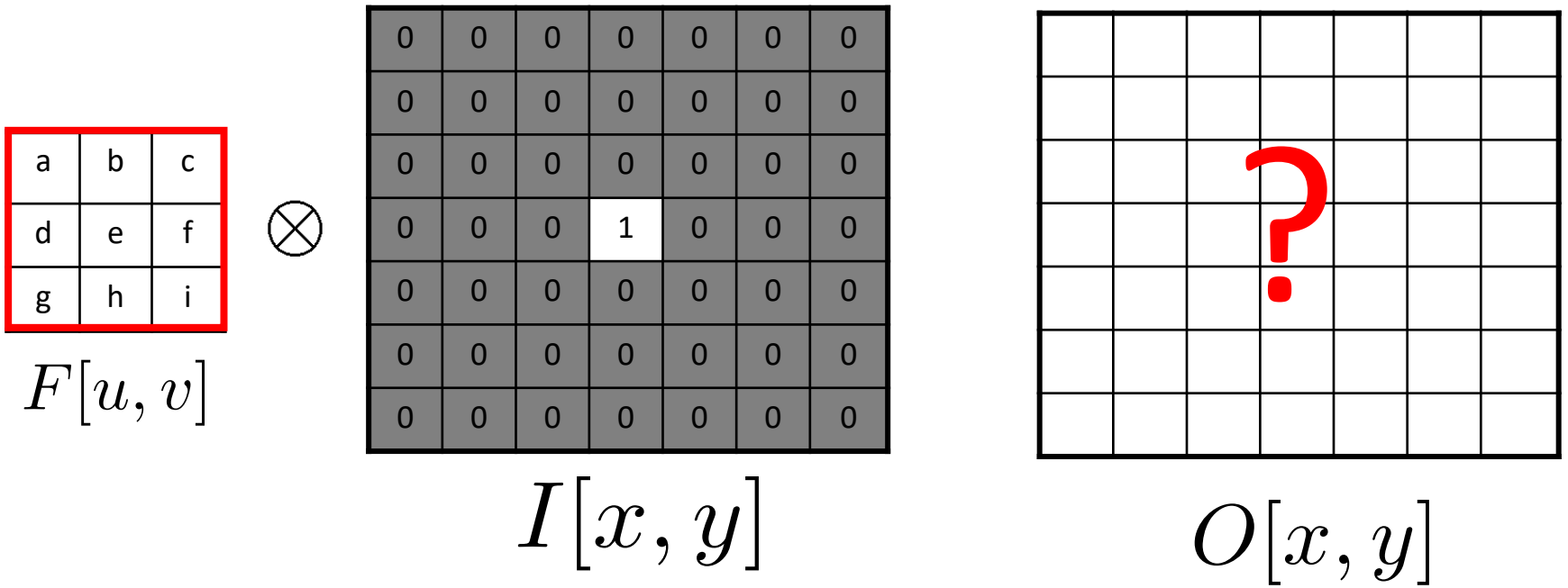
Filtrarea (liniară a) unei imagini: înlocuim fiecare pixel cu o combinație (liniară) a "vecinilor" săi.

F se mai numeşte filtru, kernel, mască (de filtrare).

Filtrul F - conține ponderile pixelilor folosiți în combinația (liniară).

Filtrarea unui semnal impuls

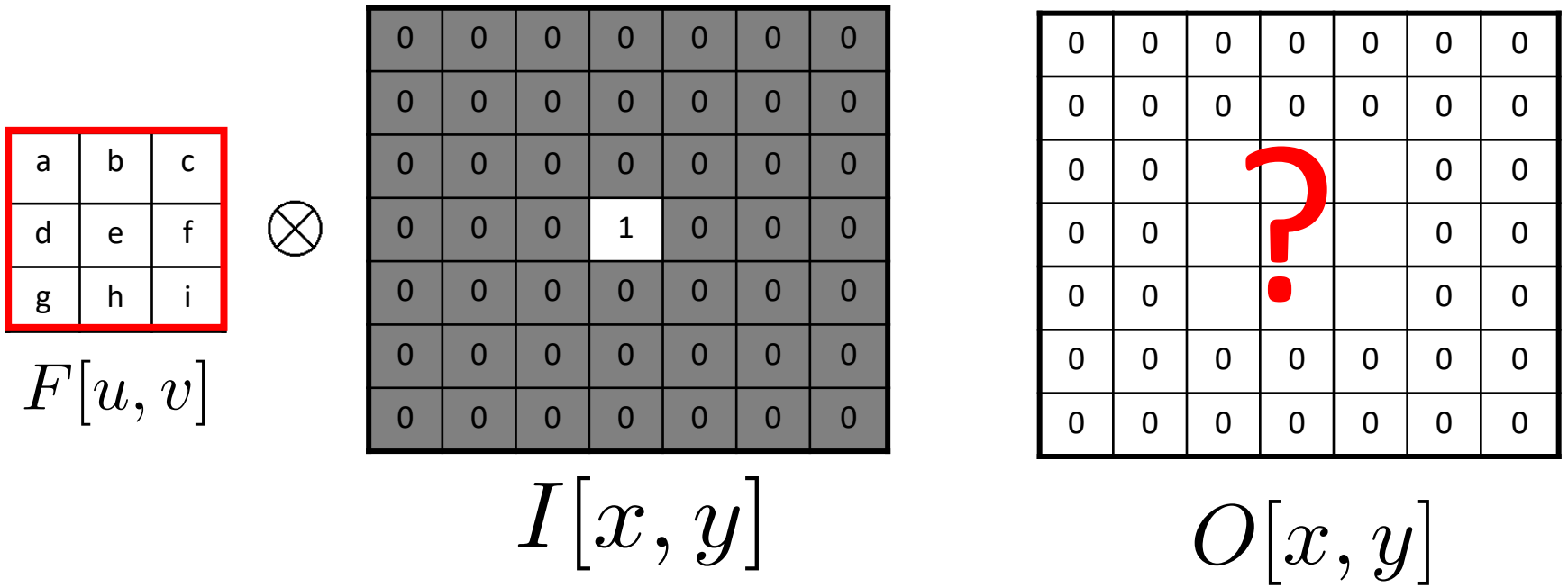
Ce rezultă în urma filtrării semnalului impuls (imaginea) I cu filtrul F ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

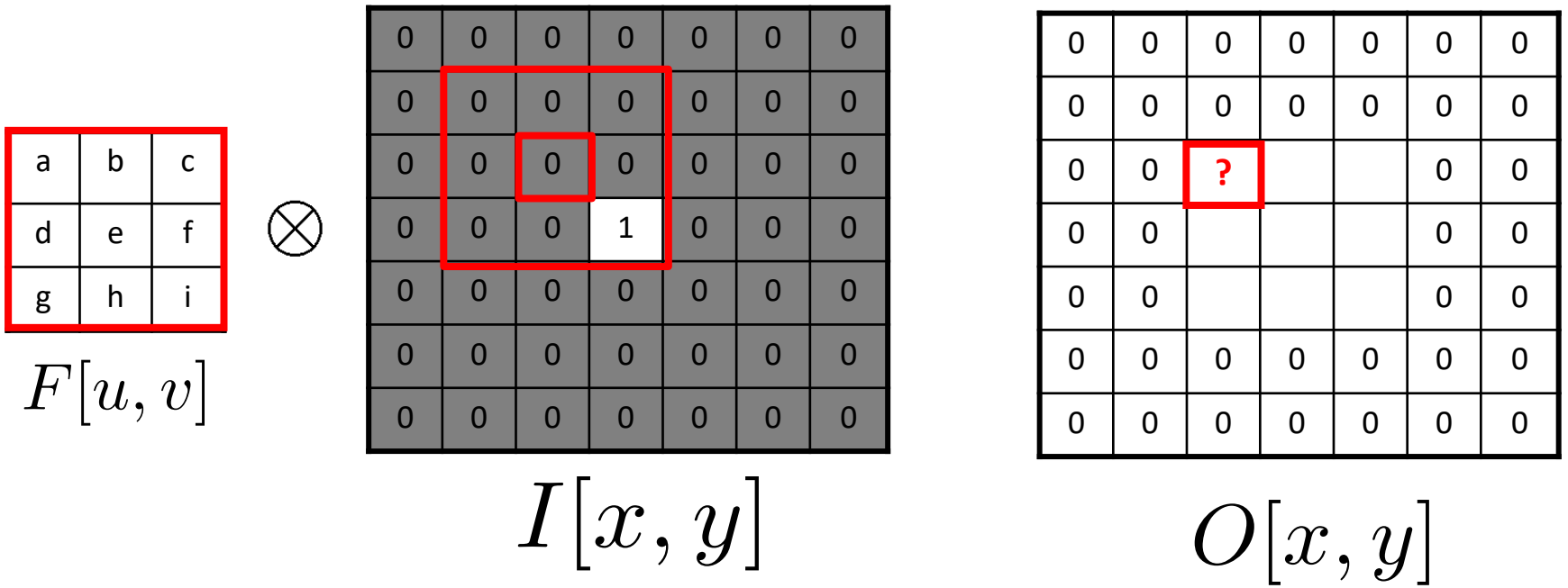
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

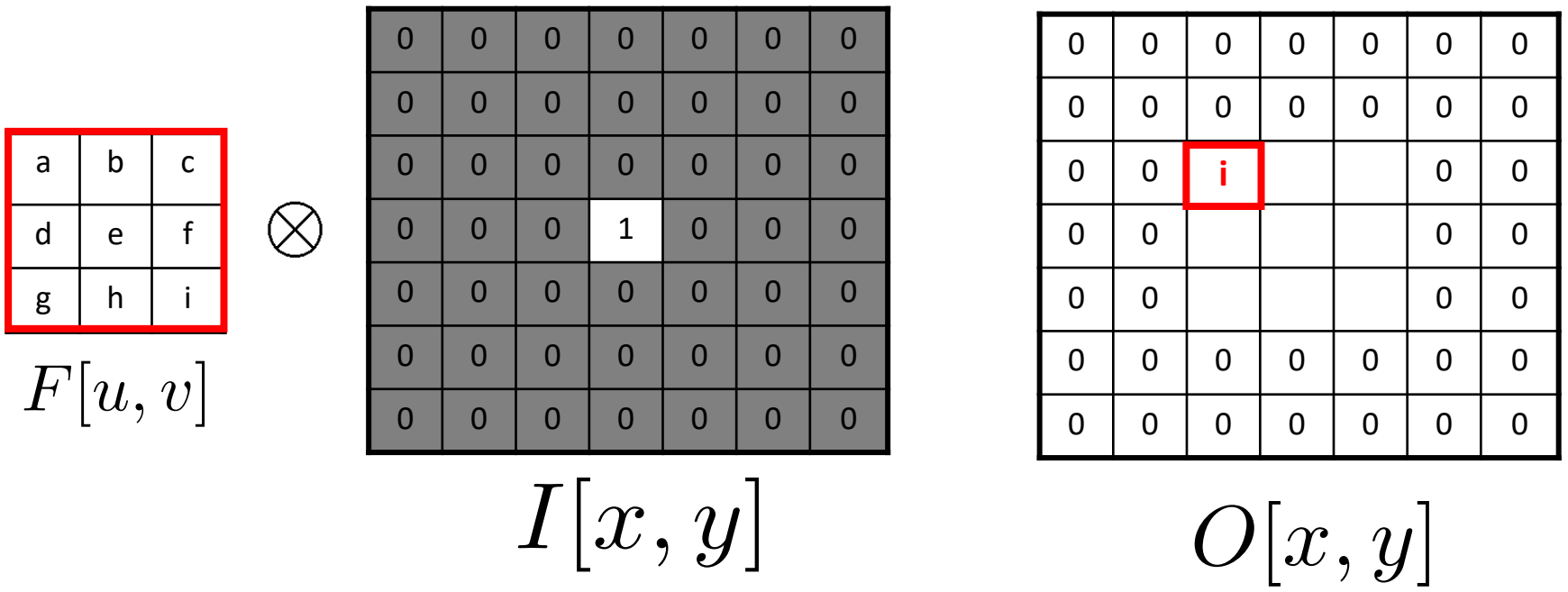
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

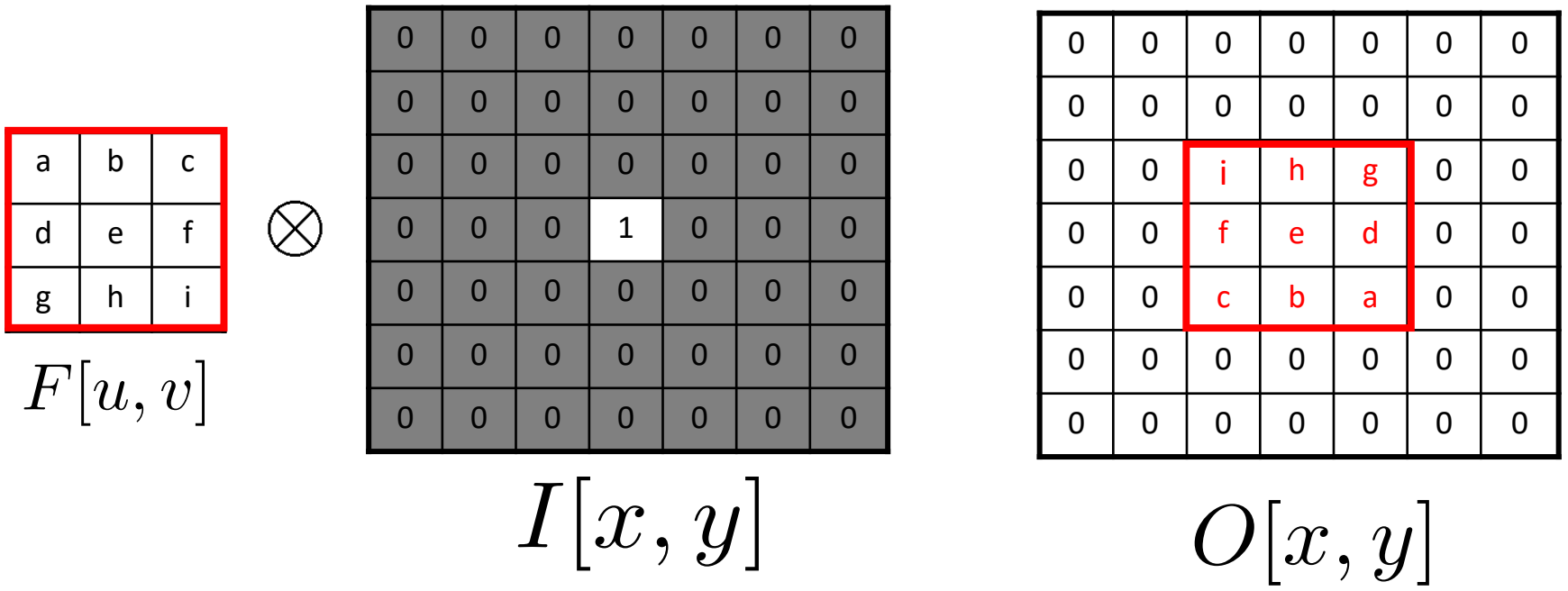
Ce rezultă în urma filtrării semnalului impuls (imaginea) *I* cu filtrul *F* ?



Slide adaptat după Steve Seitz

Filtrarea unui semnal impuls

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



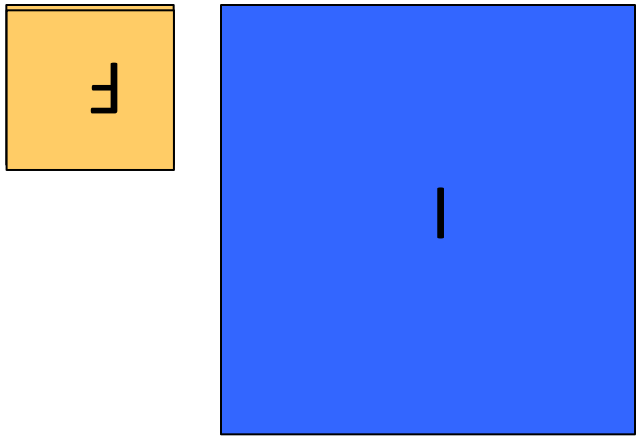
Convoluție

- 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



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

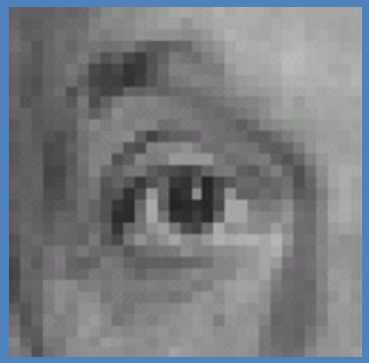
$$O[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u,v] I[i+u, j+v]$$

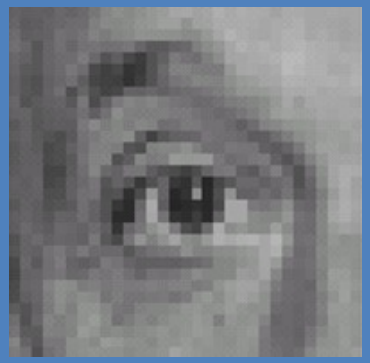
$$O = F \otimes I$$

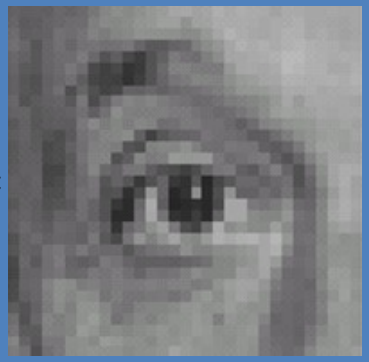
Pentru un filtru normal/de medie, cum diferă output-ul?

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

Output = ?

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

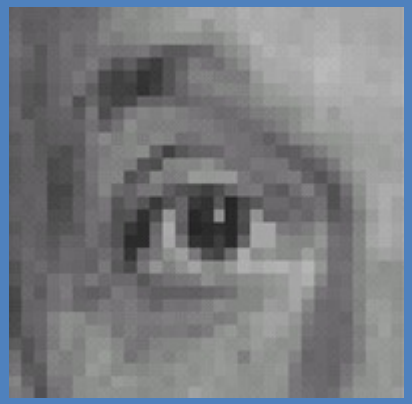
$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \text{  } = ?$$

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

Filtre liniare

0	0	0
0	1	0
0	0	0

*



=

?

input

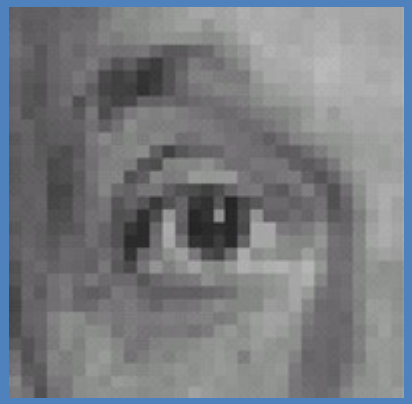
output

Slide adaptat după David Lowe

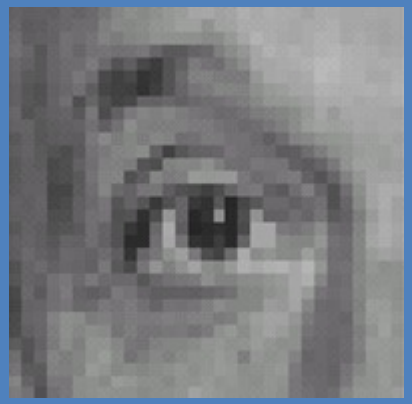
Filtre liniare

0	0	0
0	1	0
0	0	0

*



=

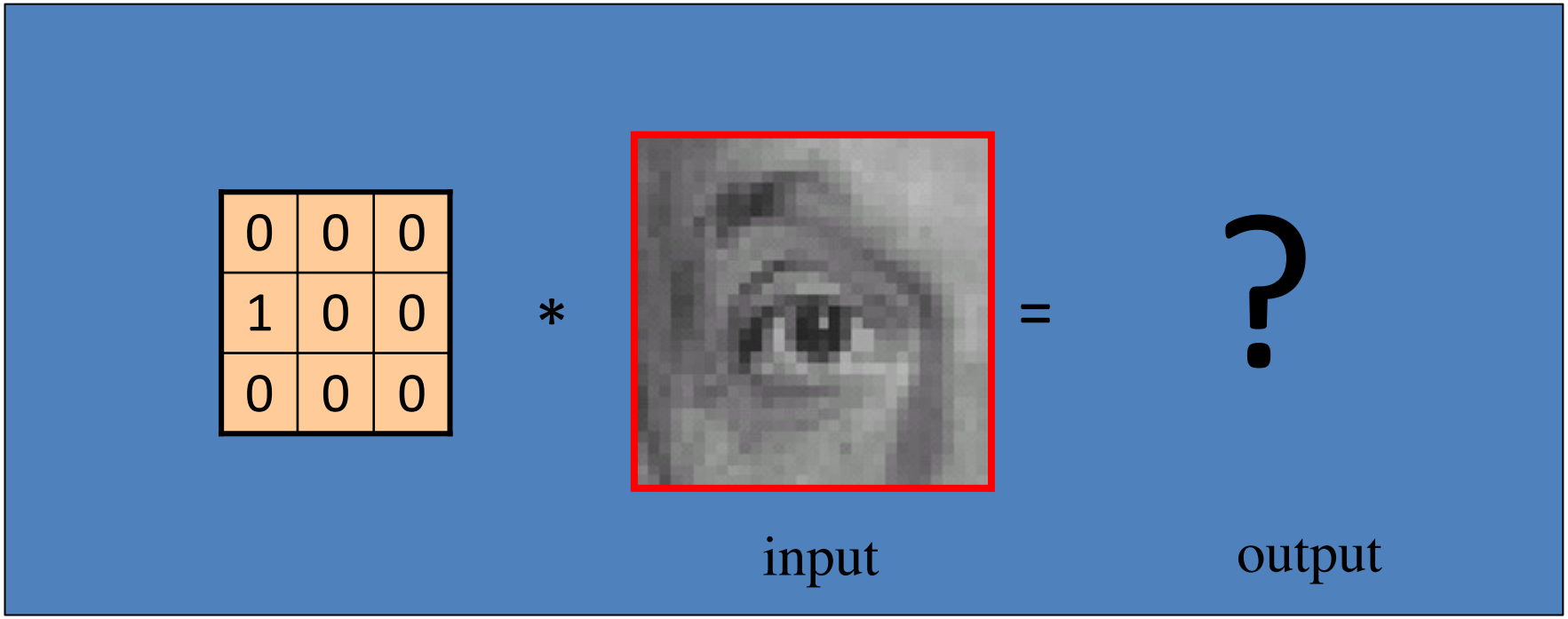


input

output (la fel)

Slide adaptat după David Lowe

Filtre liniare

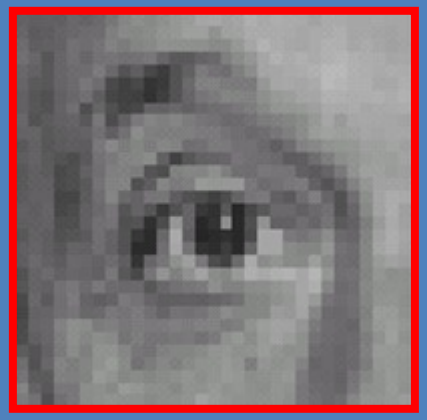


Slide adaptat după David Lowe

Filtre liniare

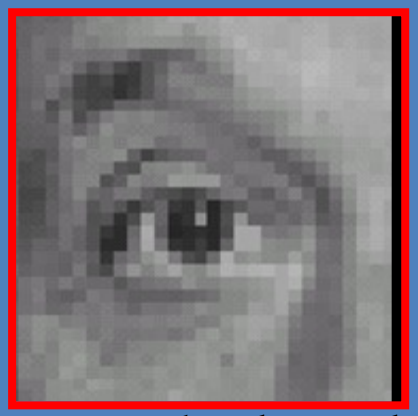
0	0	0
1	0	0
0	0	0

*



input

=



output: deplasat la stânga cu 1 pixel

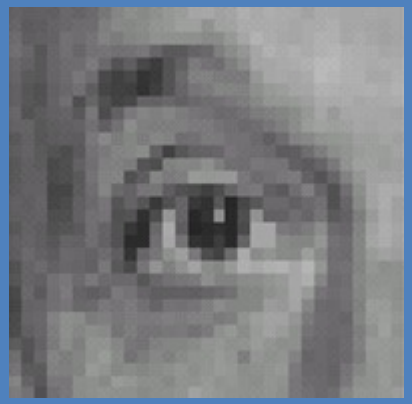
Slide adaptat după David Lowe

Filtre liniare

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

*



=

?

input

output

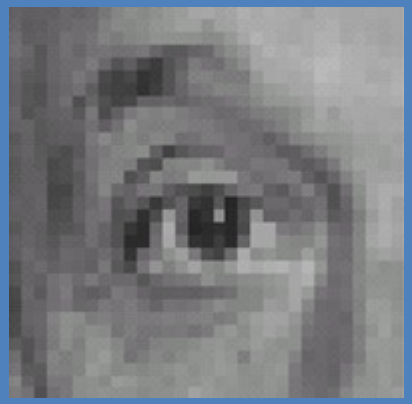
Slide adaptat după David Lowe

Filtre liniare

$\frac{1}{9}$

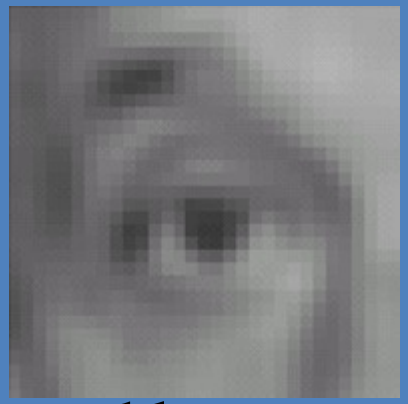
1	1	1
1	1	1
1	1	1

*



input

=

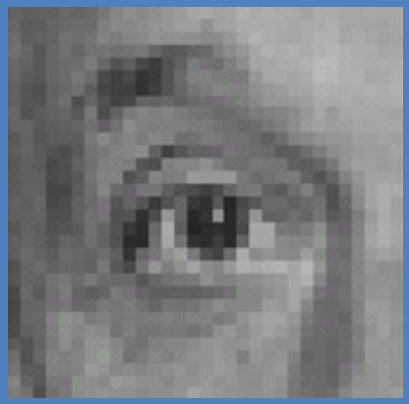


blurat
(cu filtru de medie)

Slide adaptat după David Lowe

Filtre liniare

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



(Suma = 1)

input

output

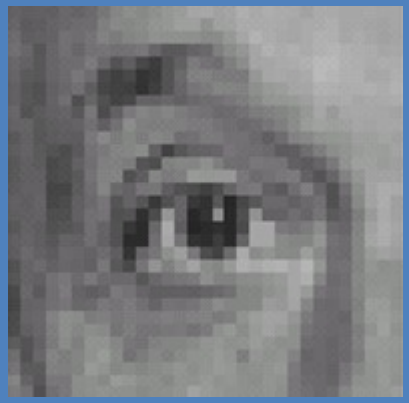
Slide adaptat după David Lowe

Filtre liniare

$\frac{1}{9}$

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

*



=

?

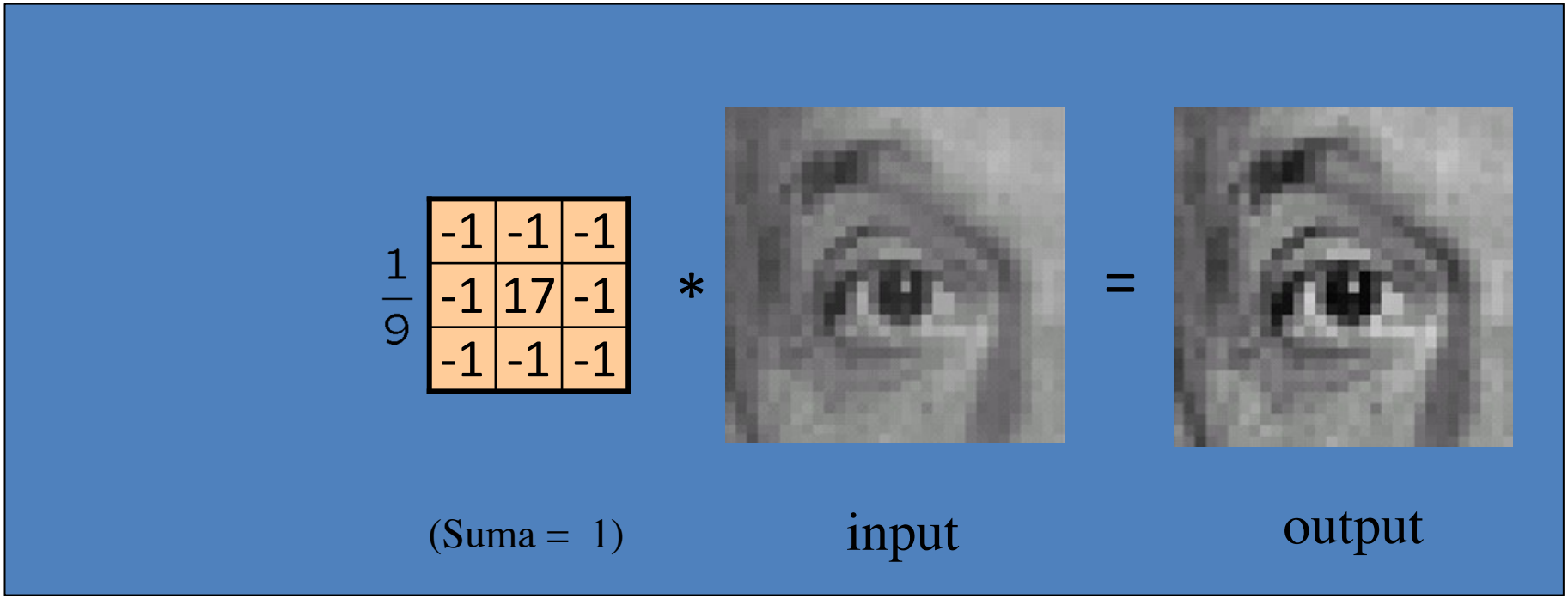
(Suma = 1)

input

output

Slide adaptat după David Lowe

Filtre liniare



Filtru de accentuare - accentuează diferențele cu ajutorul mediilor locale

Slide adaptat după David Lowe

Reducerea de zgomot

imagine originală

“salt and pepper”

impuls

normal

Reducerea zgomotului "salt and pepper"

Aplicăm un filtru Gaussian de dimensiune $n \times n$

3×3

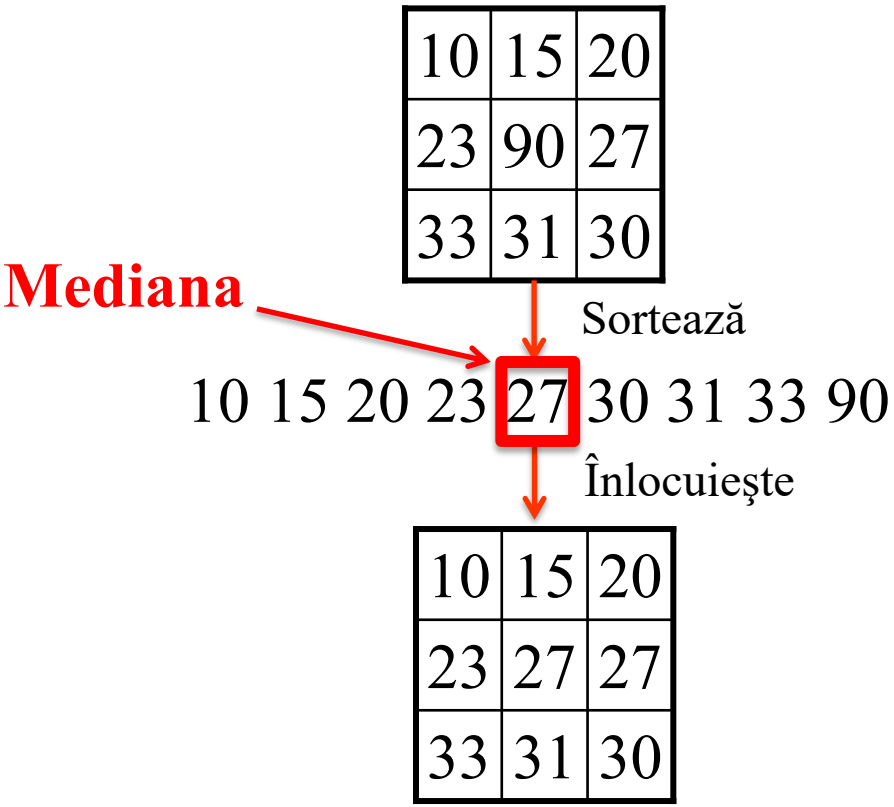
5×5

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

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

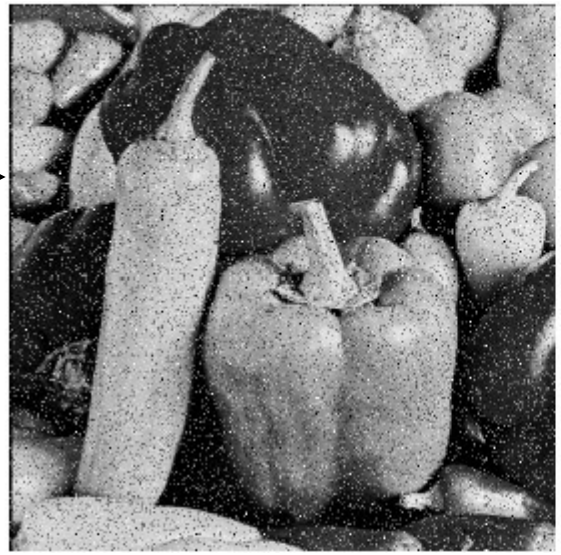


Zgomotul se elimină imaginea cu un filtru median 3 x 3. Pentru vecinătăți mai mari imaginea se deteriorează (se uniformizează).

Filtrul median

zgomot

“salt and peper”



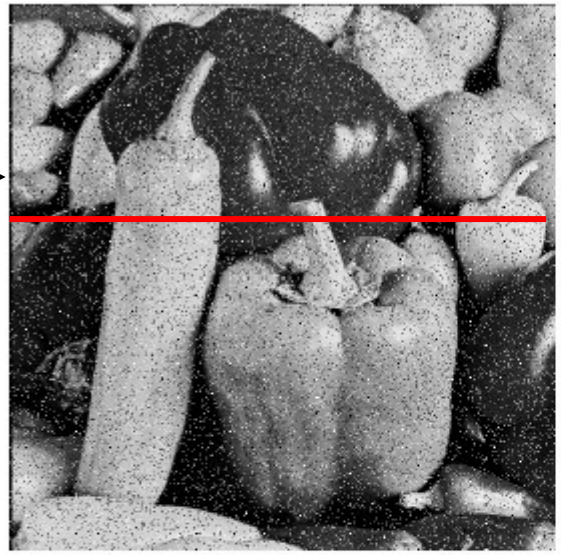
după aplicarea filtrului median

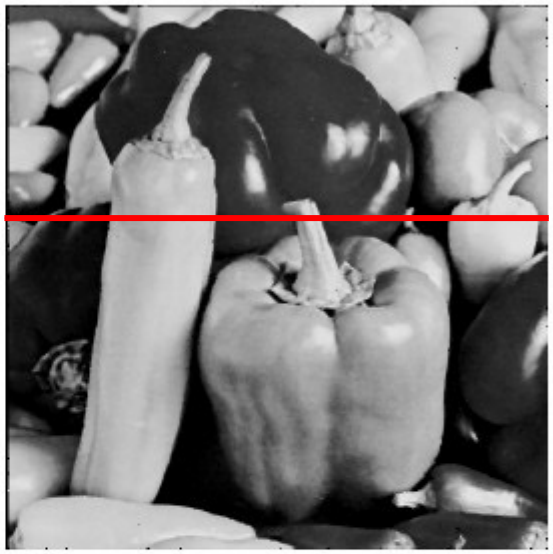
Slide adaptat după M. Herbert

Filtrul median

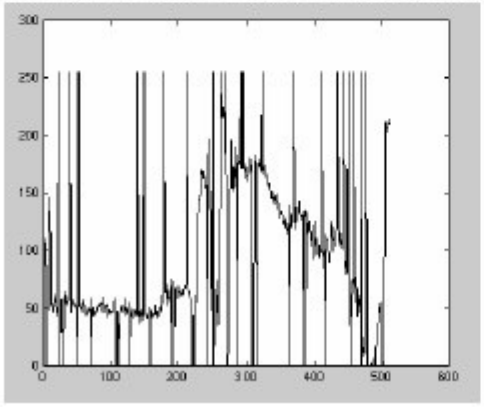
zgomot

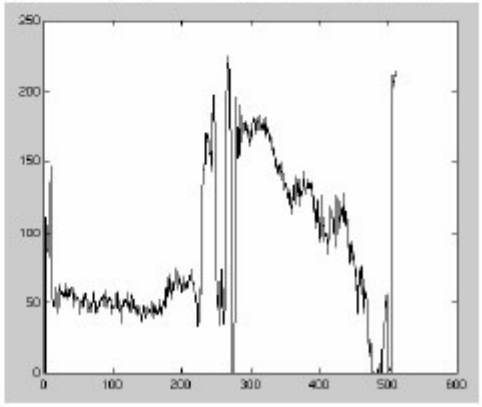
“salt and peper”





după aplicarea filtrului median





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

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)

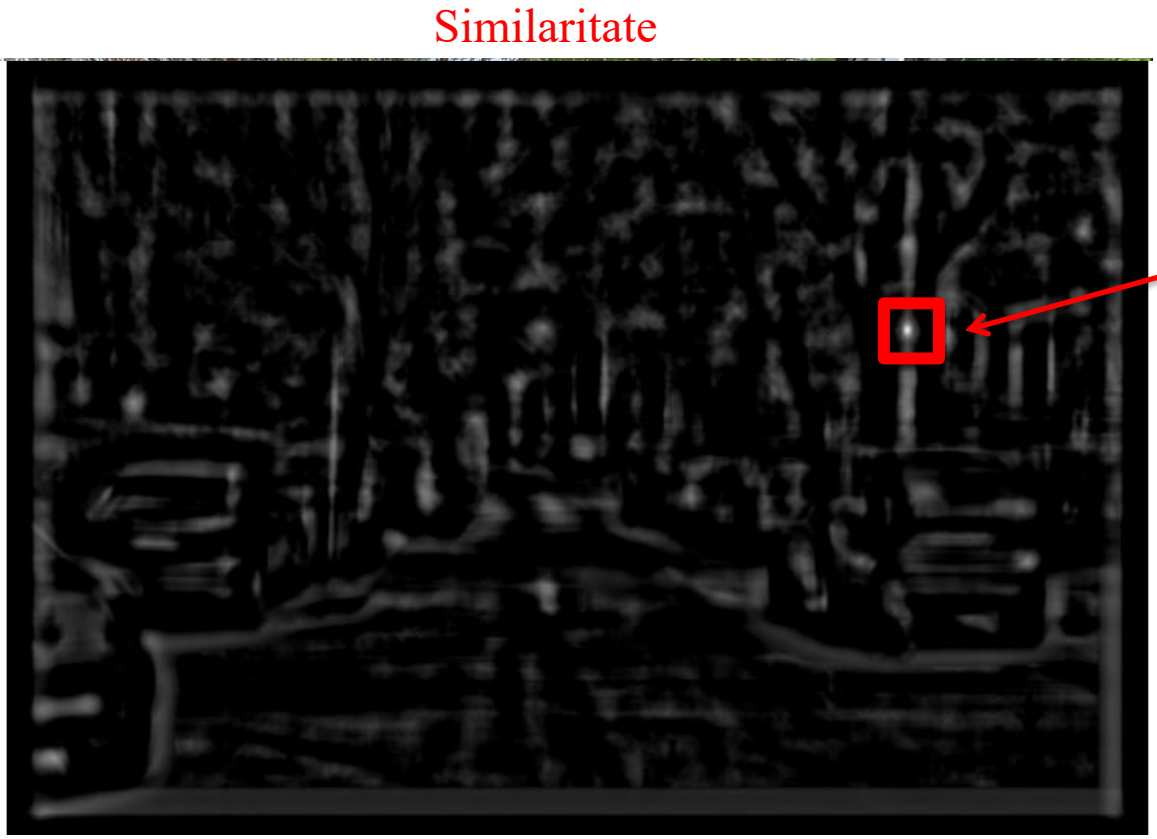
Găsirea şabloanelor - exemplu

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)

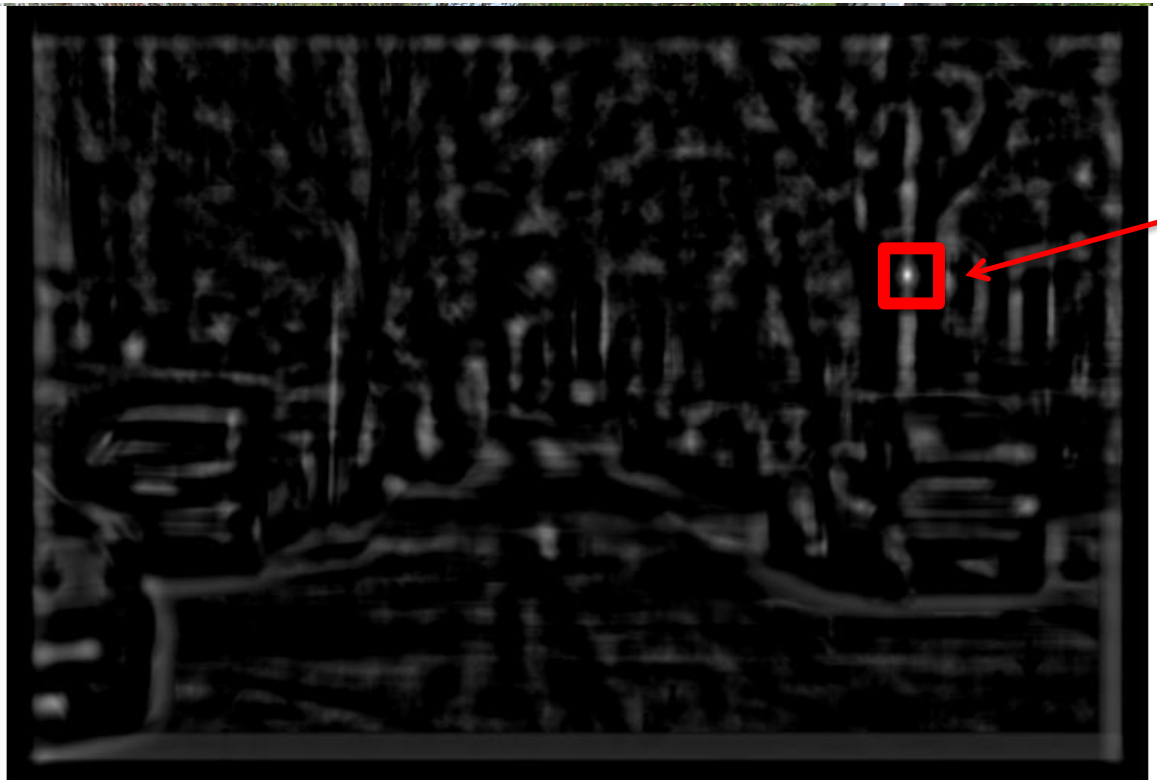
Găsirea şabloanelor - exemplu



- 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 şablonul (similaritate)

Găsirea şabloanelor - exemplu

Similaritate



Valoarea cea mai mare a similarității

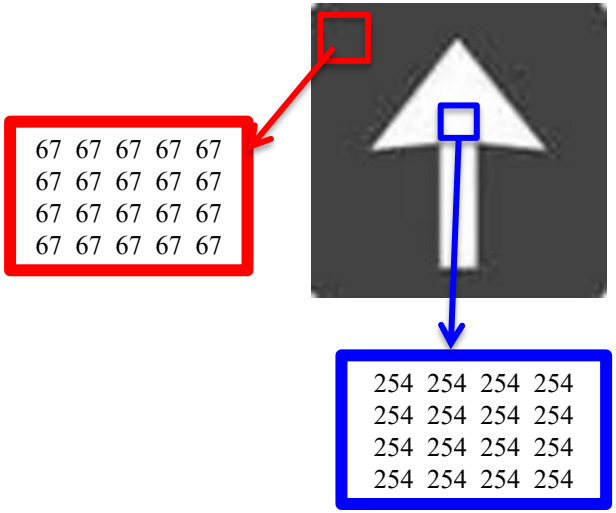
șablon
(sens unic)

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

Găsirea şabloanelor cu filtre

- şablon =

rgb2gray
cv.cvtColor



rgb2gray = cv.cvtColor

Slide adaptat după Derek Hoiem

Găsirea şabloanelor cu filtre

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

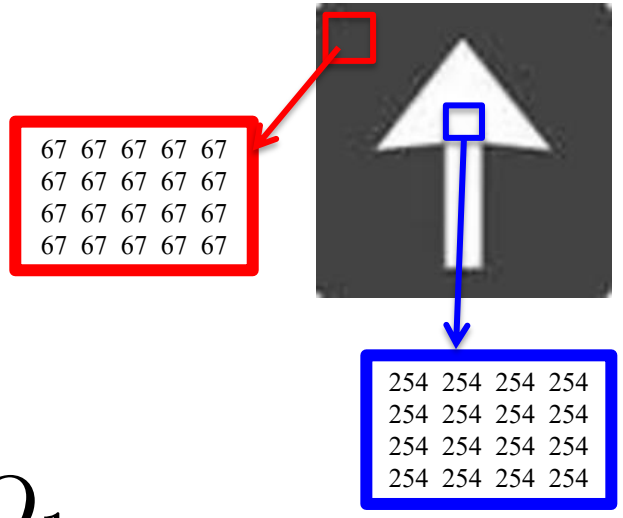
$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_1(u,v) I(i+u, j+v)$$

imagine complet albă

O_1

Explicaţii?

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)
- corelaţie: 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, $f_3 = f_2 - \overline{f_2}$)
- corelaţie: 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

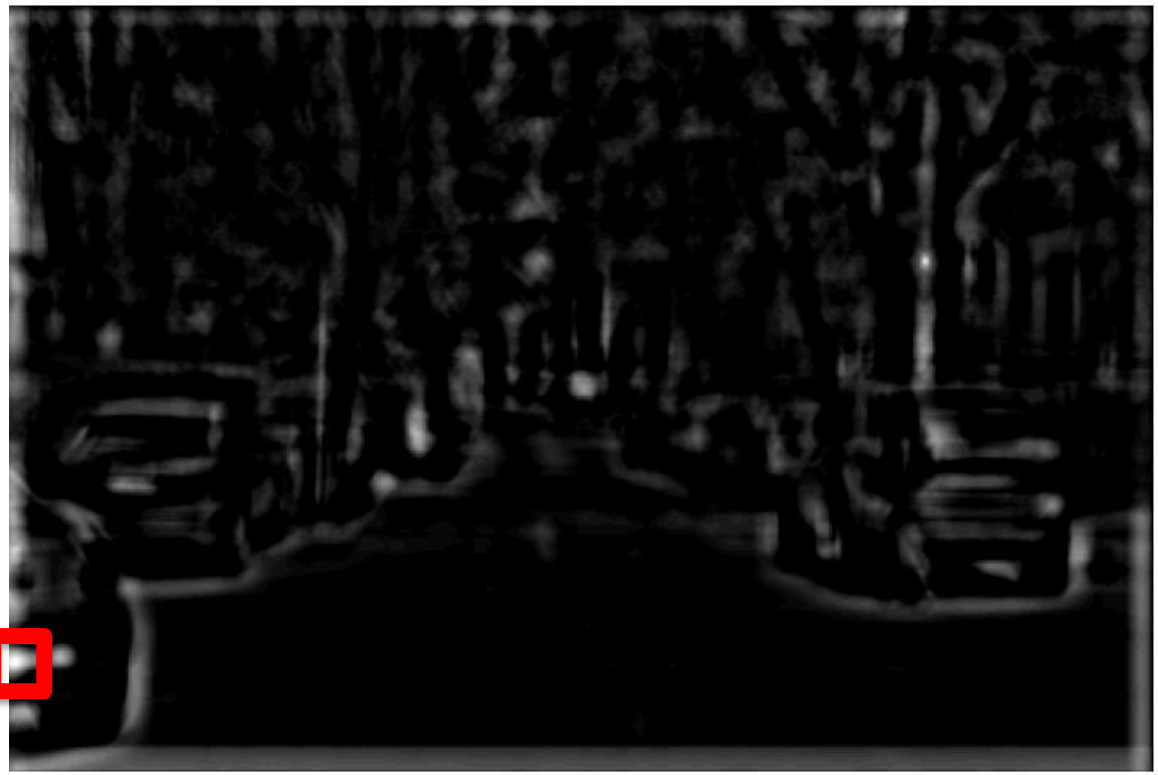
$$O_3 = f_3 \otimes I$$

Ce rezultă după filtrare?

Găsirea şabloanelor cu filtre

- filtrul f_3 =  normalizat (suma = 0, $f_3 = f_2 - \overline{f_2}$)
- corelaţie: 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

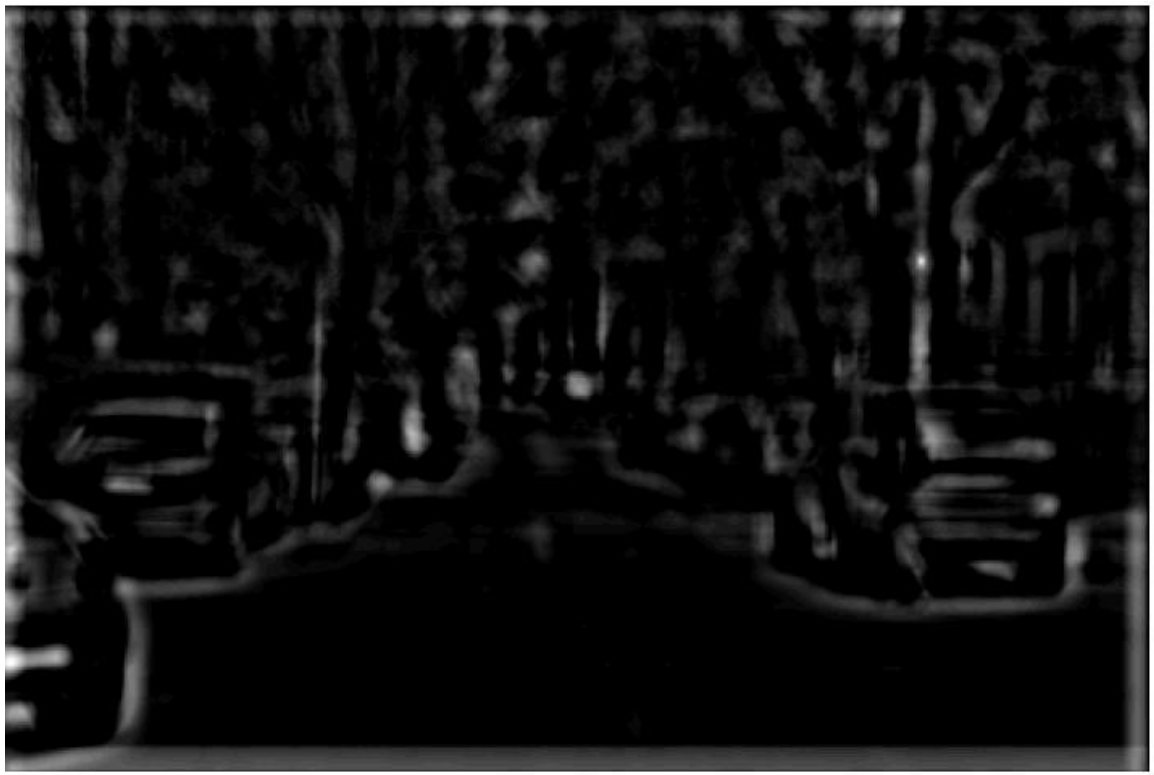
Explicaţii?

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

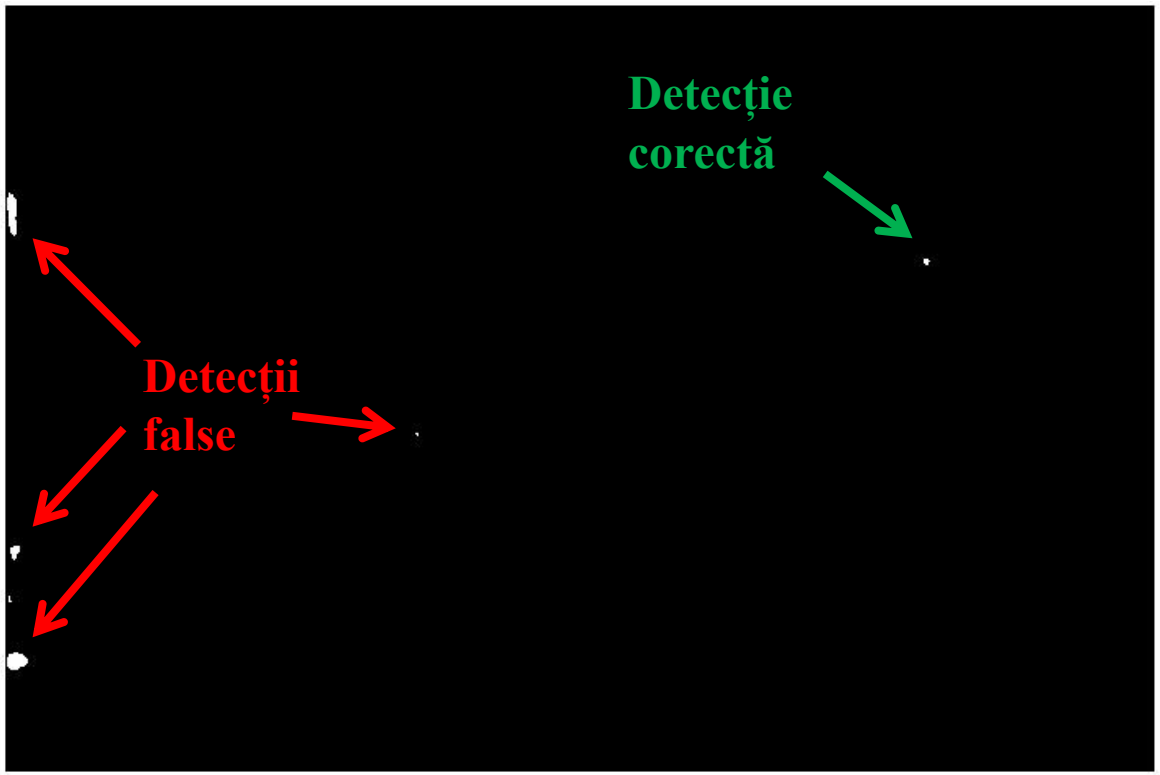
Explicații?

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

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

Găsește valoarea minimă a distanței/maximă a similarității detectând șablonul.

O_4 (1 – distanta)

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 (1 – distanta)

Găsirea şabloanelor cu filtre

- filtrul f_1 = 
- corelaţie normalizată (de medie 0)

res = cv.matchTemplate(img,template,method)

media intensităţilor în ferestra curentă din imagine

media intensităţilor filtrului

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

deviaţia standard a intensităţilor în ferestra curentă din imagine

deviaţia standard a intensităţilor în filtru

Cosinusul a doi vectori normalizaţi

Găsirea şabloanelor cu filtre

- filtru $f_1 =$ 
- corelaţie normalizată

res = cv.matchTemplate(img,template,method)

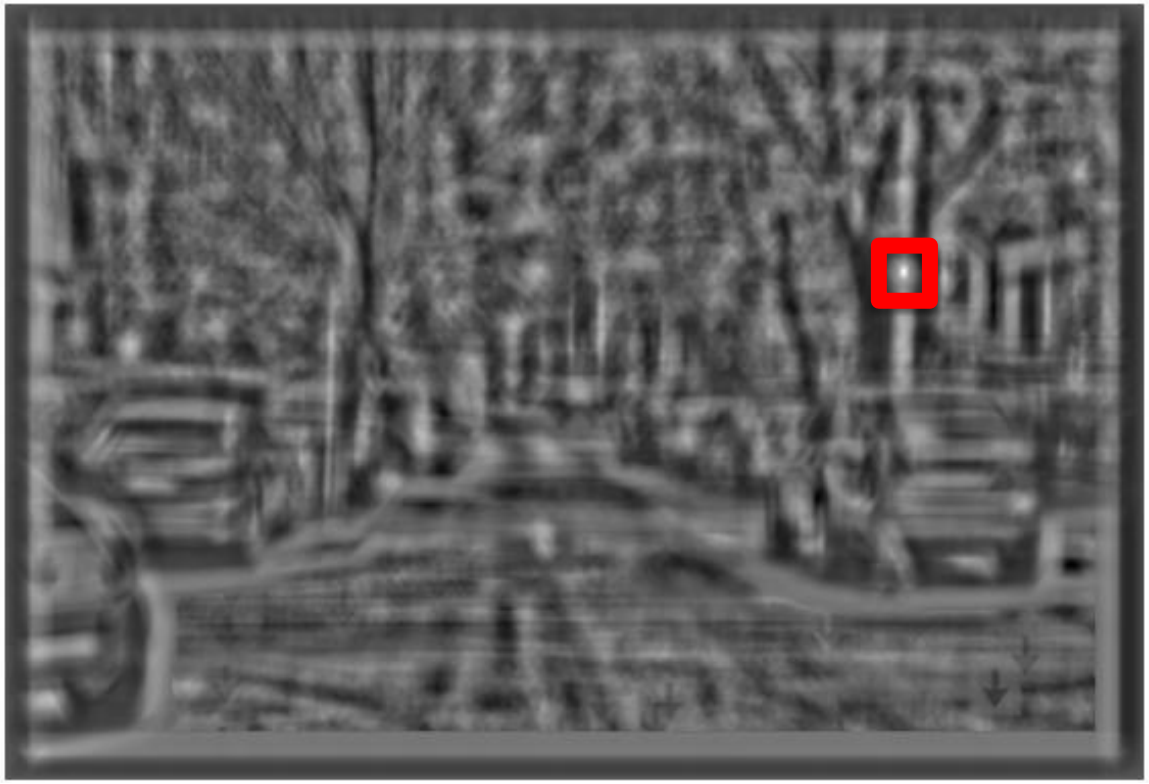
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$

Găsirea şabloanelor cu filtre

- filtru $f_1 =$ 
- corelaţie normalizată

res = cv.matchTemplate(img,template,method)

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



Găseşte valoarea minimă detectând şablonul.

Găsirea şabloanelor cu filtre

- filtru $f_1 =$ 
- corelaţie normalizată

res = cv.matchTemplate(img,template,method)

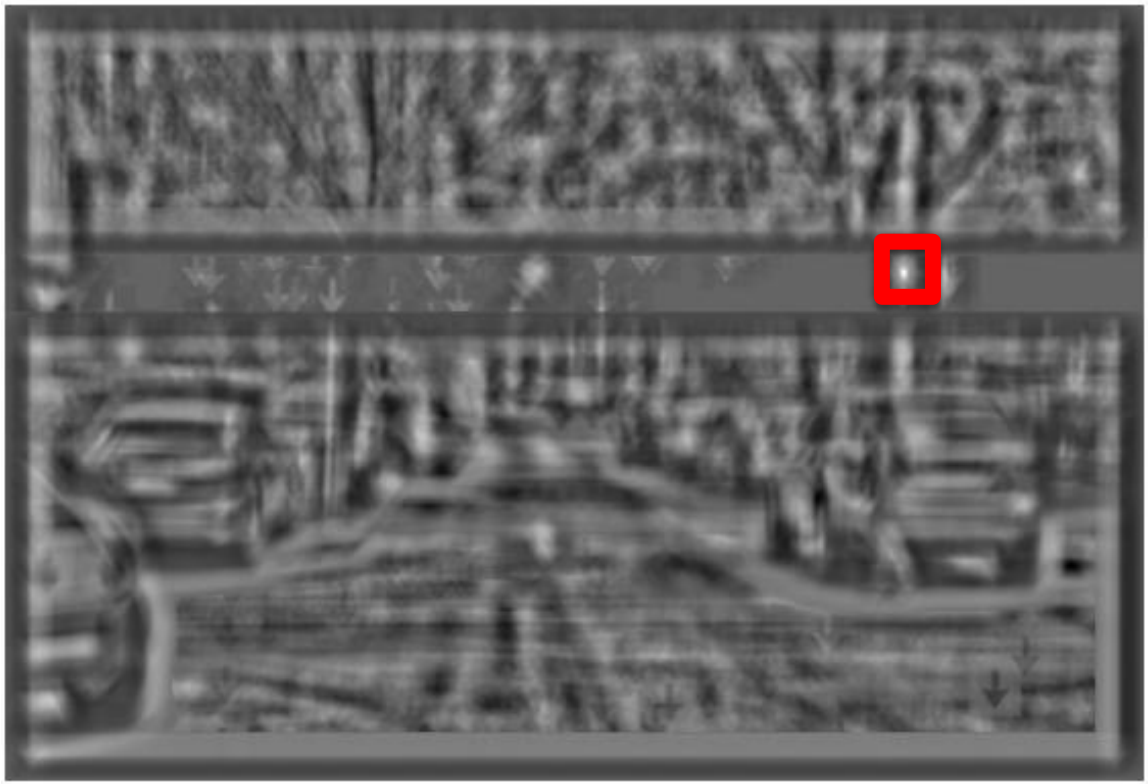
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$

Găsirea şabloanelor cu filtre

- filtru $f_1 =$ 
- corelaţie normalizată

res = cv.matchTemplate(img,template,method)

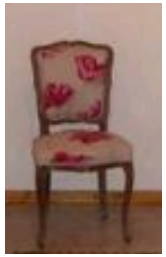
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})(f_1-\overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v)-\overline{I}_{u,v})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1-\overline{f_1})^2}}$$



Găseşte valoarea minimă detectând şablonul.

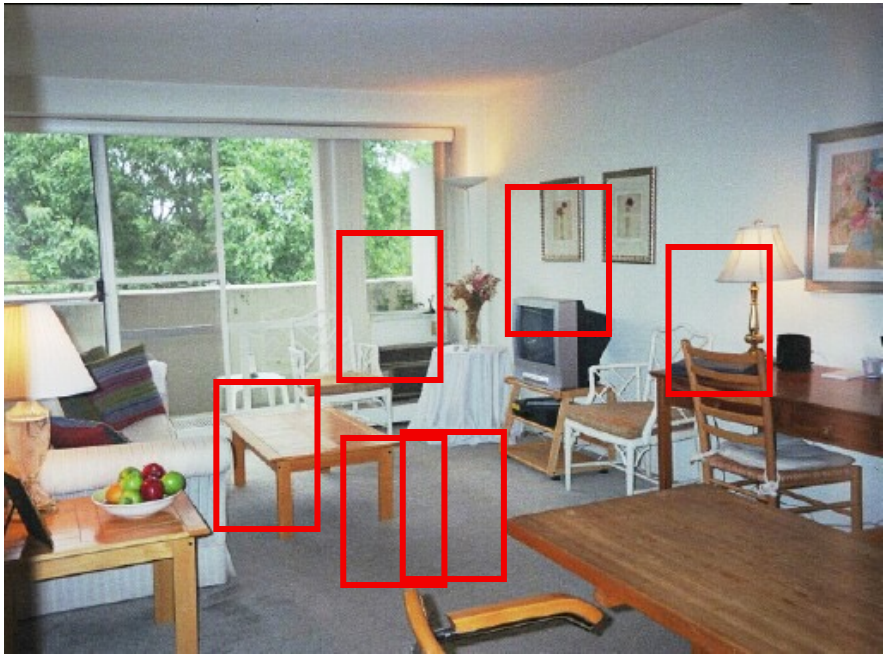
Cea mai bună măsură a similarității?

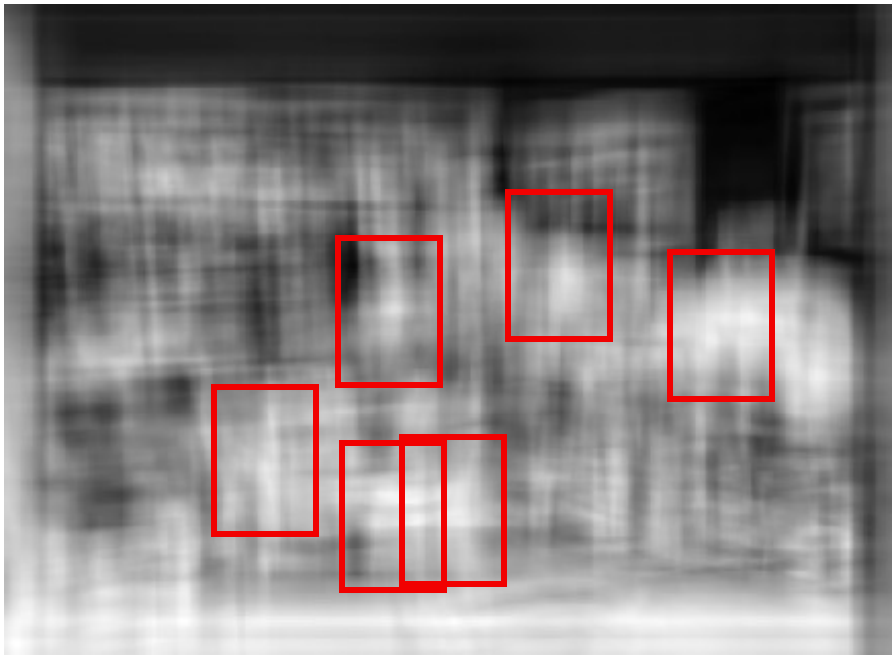
- filtru de medie 0: rezultate nu prea bune (apar detecții false)
- suma pătratelor distanțelor: senzitiv la intensitatea medie
- corelație normalizată: invariant la intensitatea medie şi la contrast



Detectare de obiecte cu găsirea şabloanelor

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

- nu este 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?

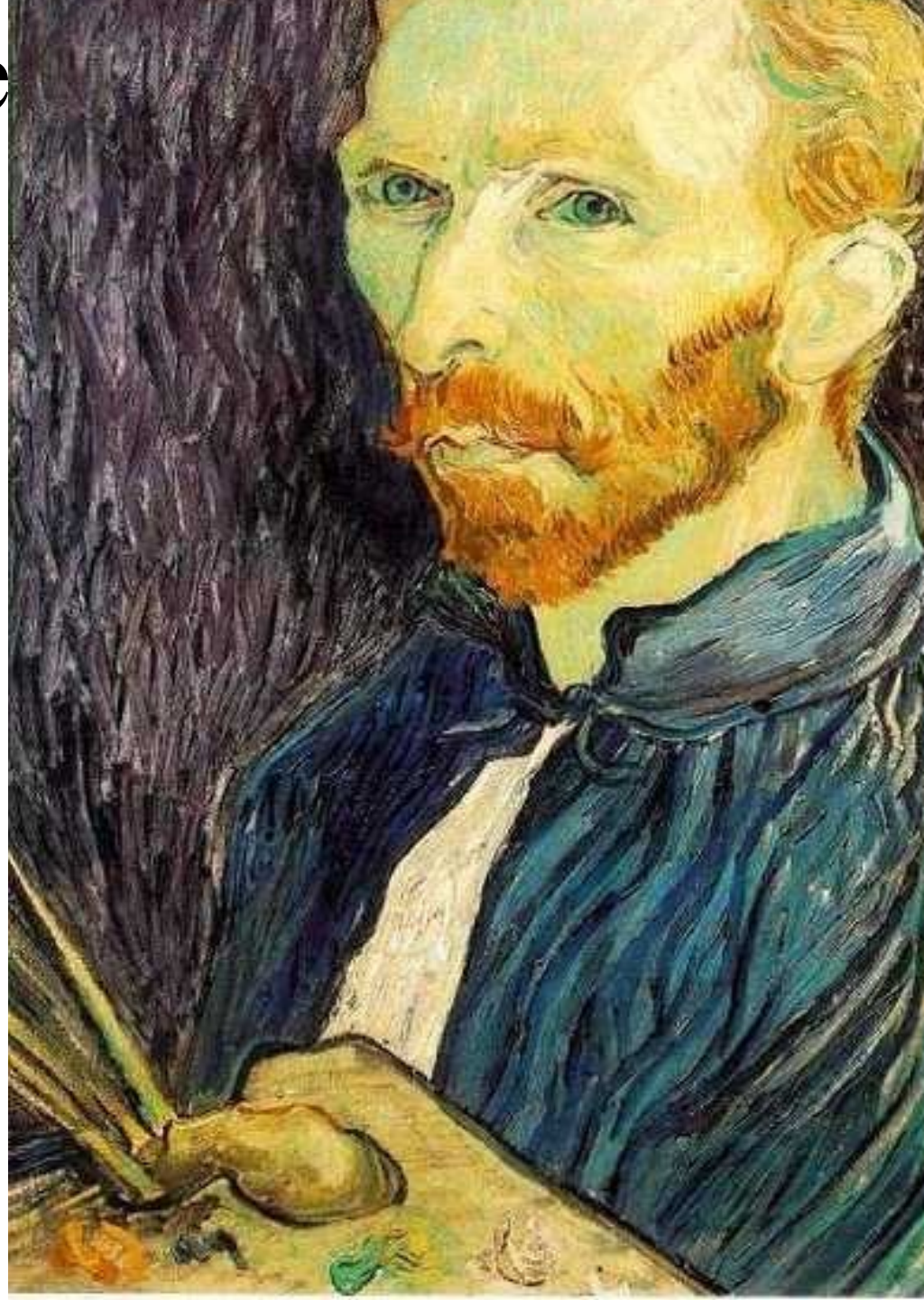
şablon iniţial 

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?



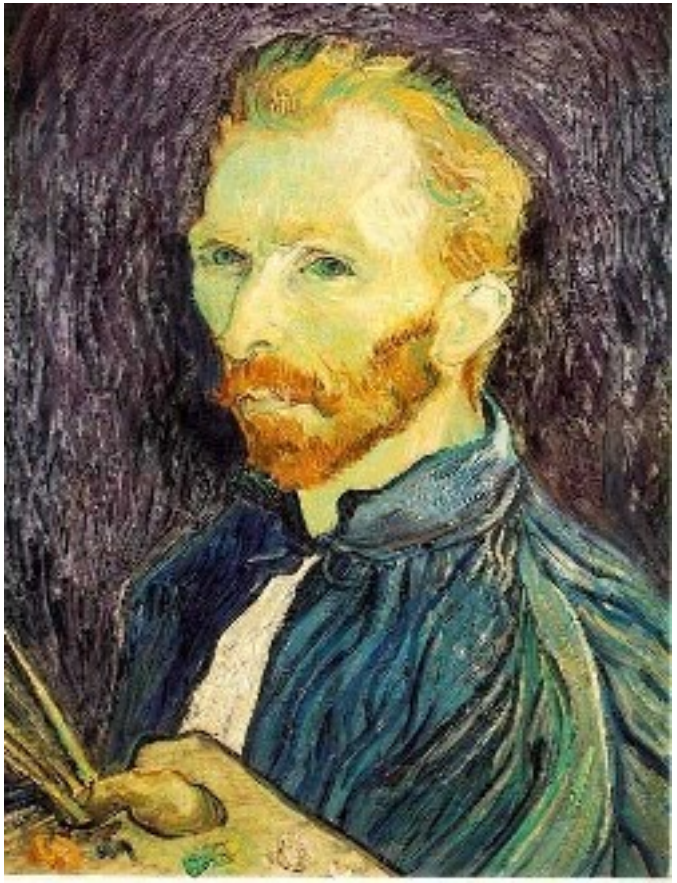
Slide adaptat după Steve Seitz

Algoritm pentru micşorarea imaginilor cu factor 2

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

```
imgRedusa = img [0::2, 0::2]
```

Rezultate



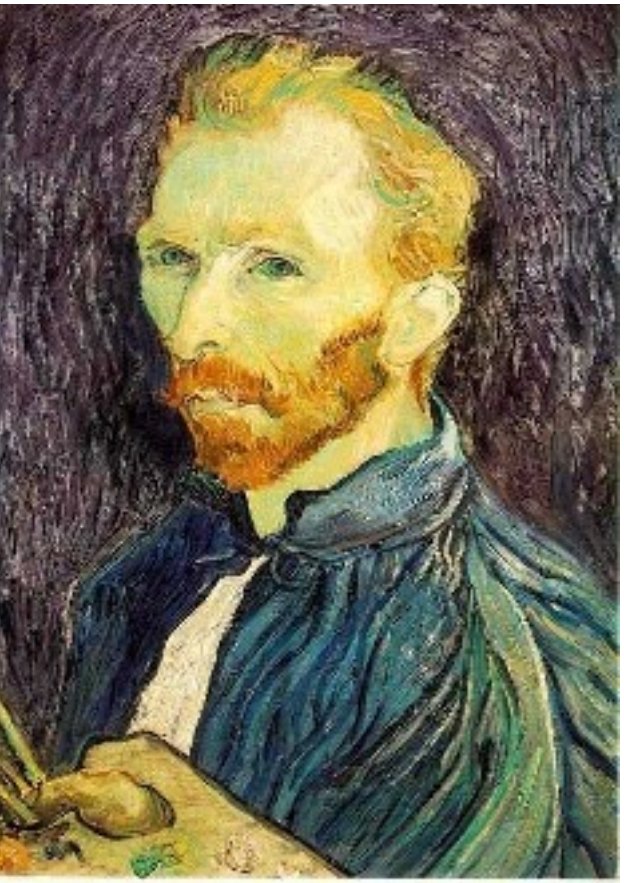
Eliminăm fiecare a doua linie şi a doua coloană pentru a crea o imagine de 2x mai mică

1/4

1/8

Slide adaptat după Steve Seitz

Rezultate - zoom



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Observații?

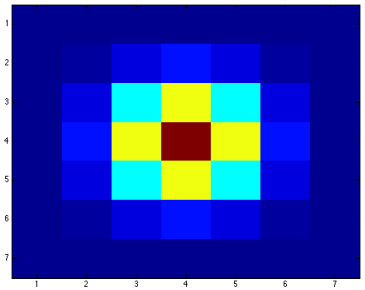
Slide adaptat după Steve Seitz

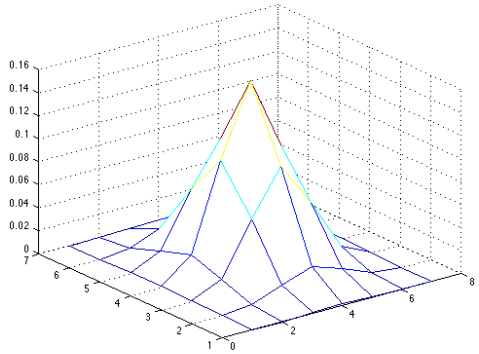
Algoritm pentru micşorarea imaginilor cu factor 2 cu filtrare Gaussiană

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

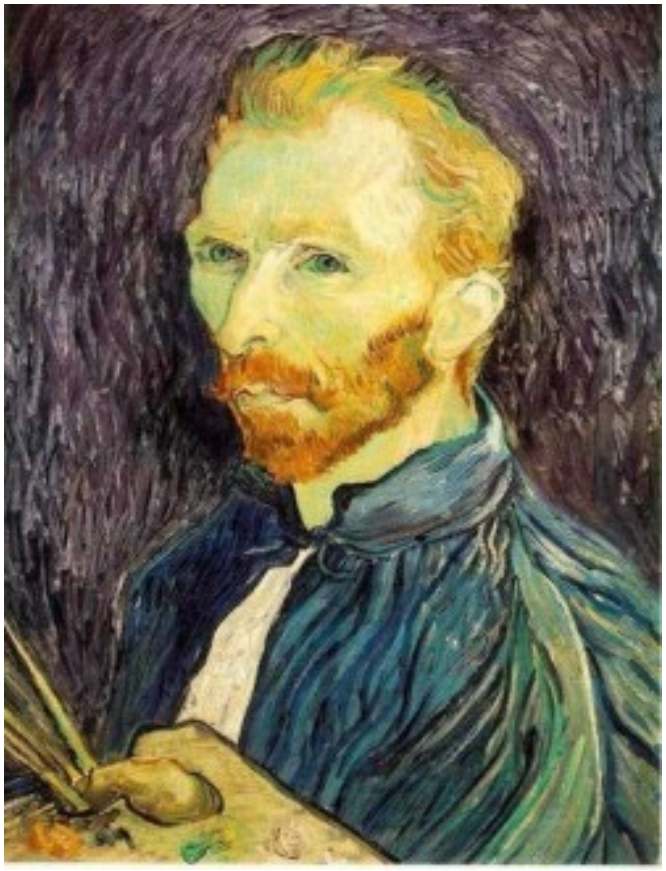
   imgBlurata = cv2.GaussianBlur(img,(5,5),0)
3. Selectează fiecare al doilea pixel

   imgRedusa = imgBlurata [0::2, 0::2]





Rezultate



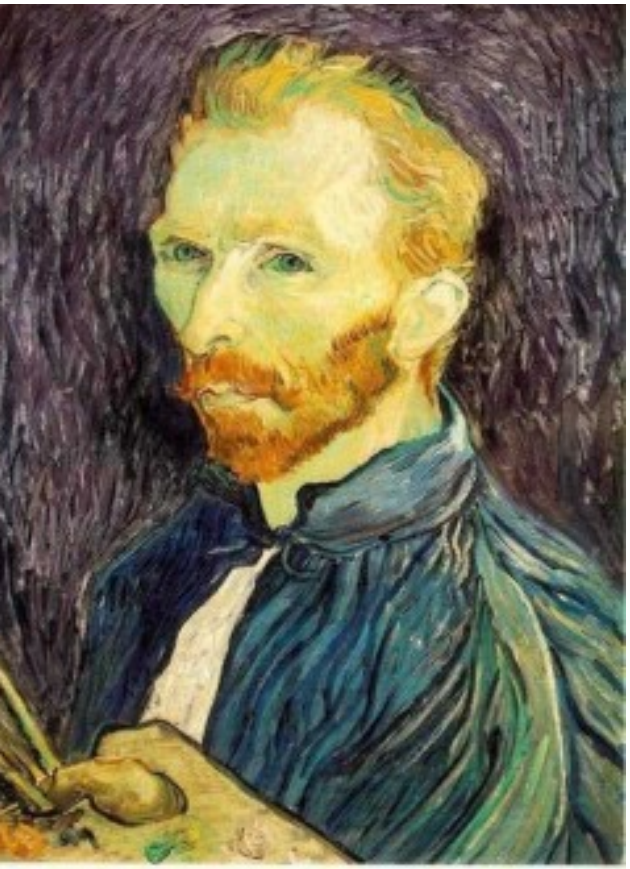
Gaussian 1/2

G 1/4

G 1/8

Slide adaptat după Steve Seitz

Rezultate - zoom



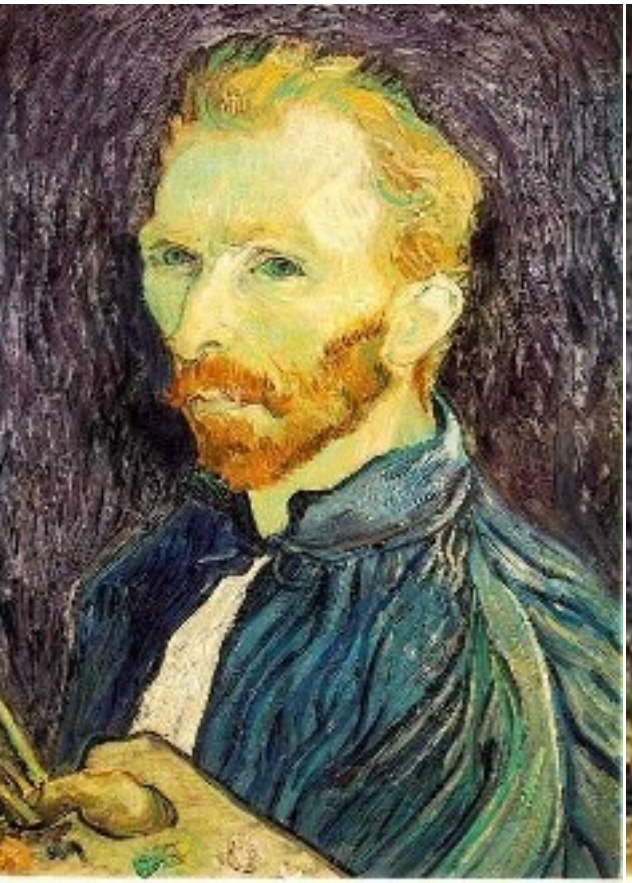
Gaussian 1/2

G 1/4

G 1/8

Slide adaptat după Steve Seitz

Versus...



1/2

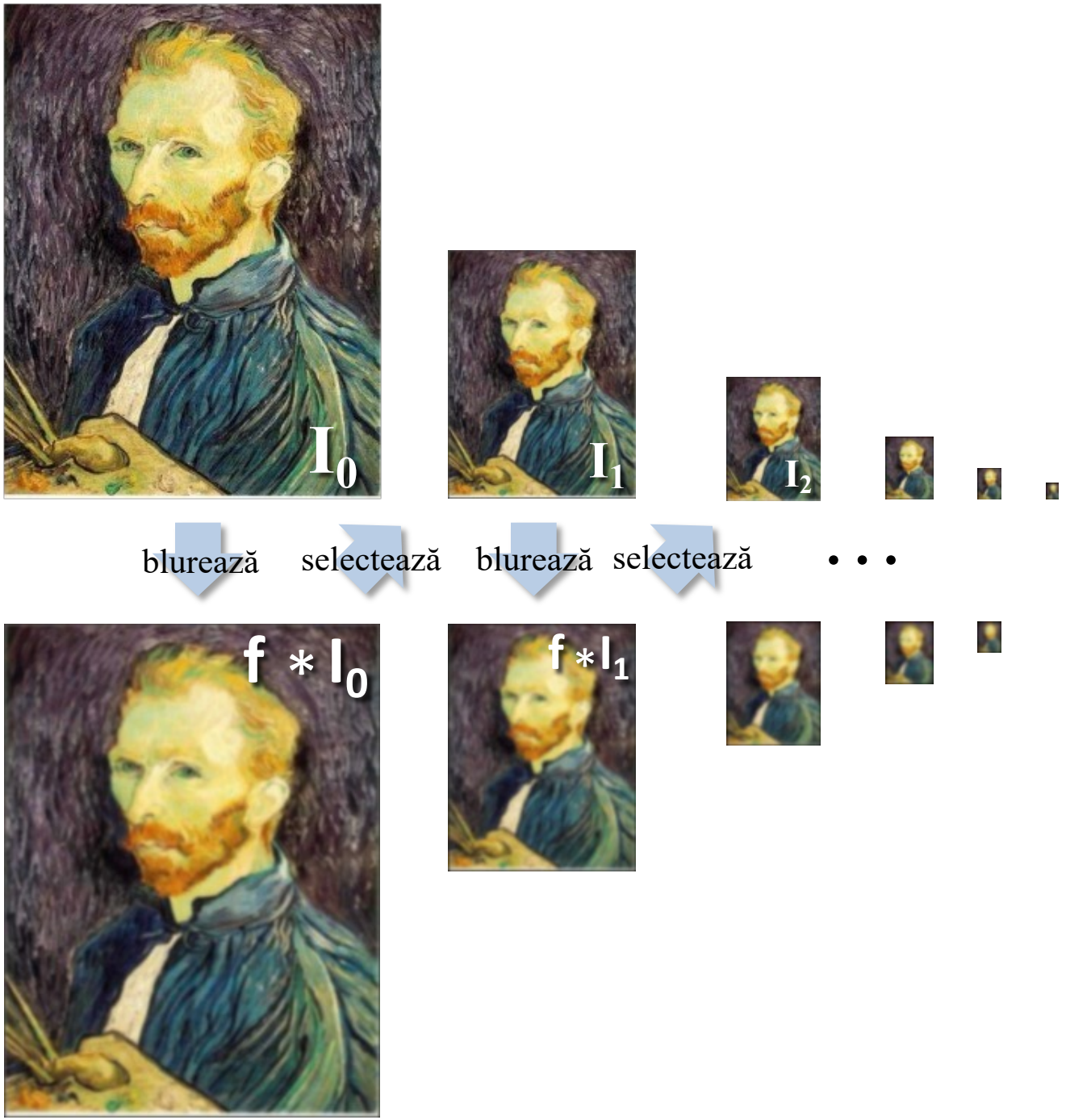
1/4 (2x zoom)

1/8 (4x zoom)

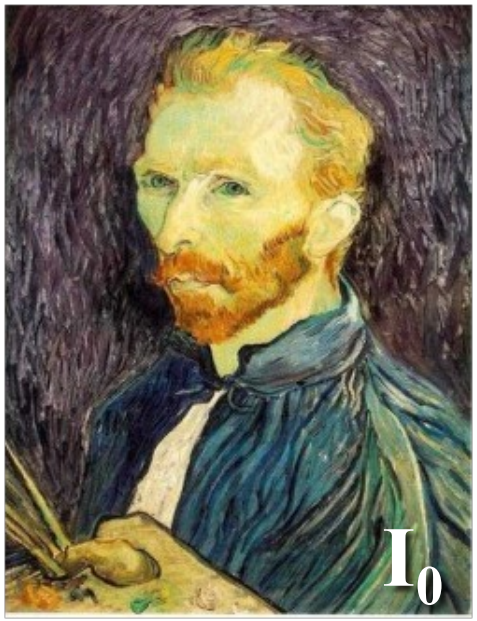
Slide adaptat după Steve Seitz

Filtrare Gaussiană

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



Piramidă Gaussiană

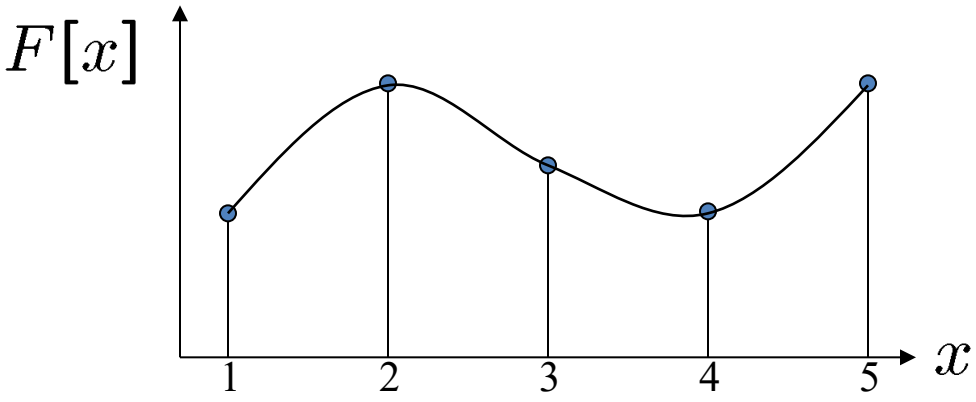


blur selectează blur selectează • • •

Mărirea imaginilor

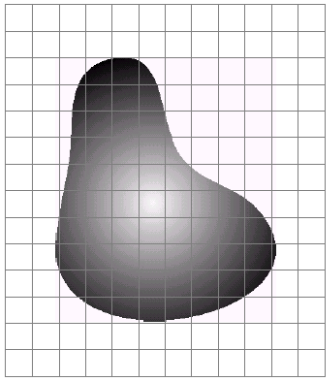
- imagine prea mică pentru ecran: 
  (45 x 45 pixeli)
- cum putem să o mărim de 10 ori?
  (450 x 450 pixeli)
- metodă simplă: repetăm fiecare rând şi coloană de 10 ori (fiecare pixel multiplicat de 100 de ori)
- interpolare “cel mai apropiat vecin”

Interpolarea imaginilor

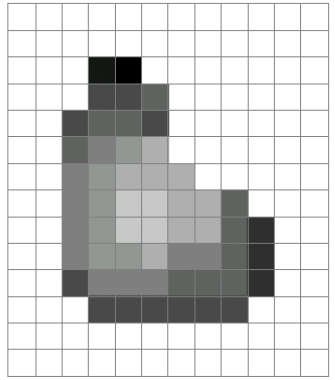


Cum se formează imaginile digitale?

- 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



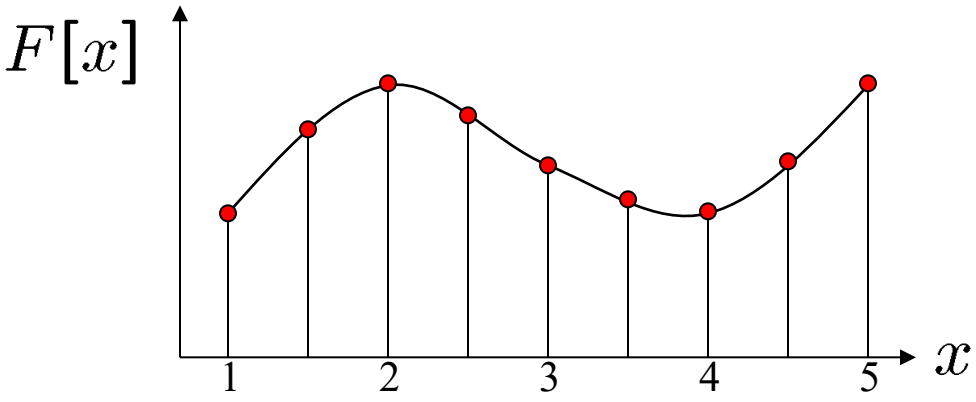
f - funcție continuă

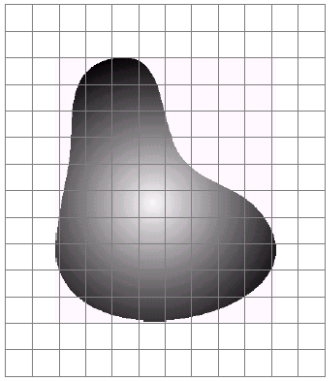


F - funcție discretă

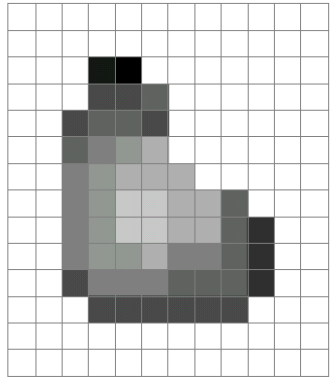
Slide adaptat după Steve Seitz

Interpolarea imaginilor





f - funcție continuă



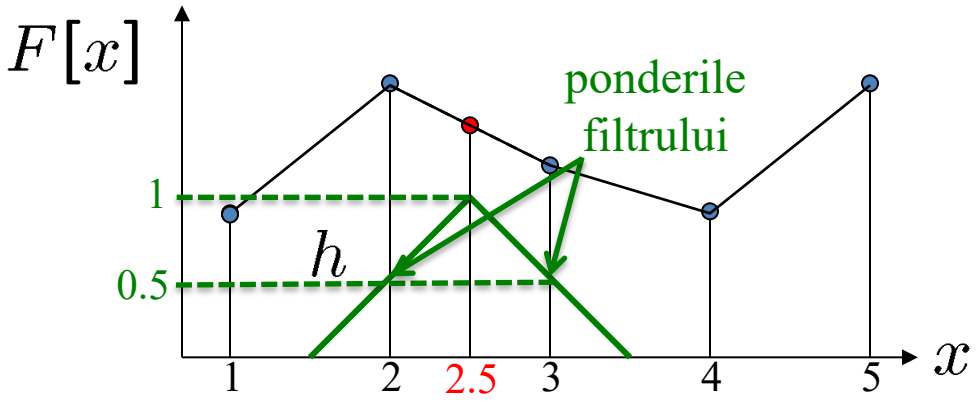
F - funcție discretă

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

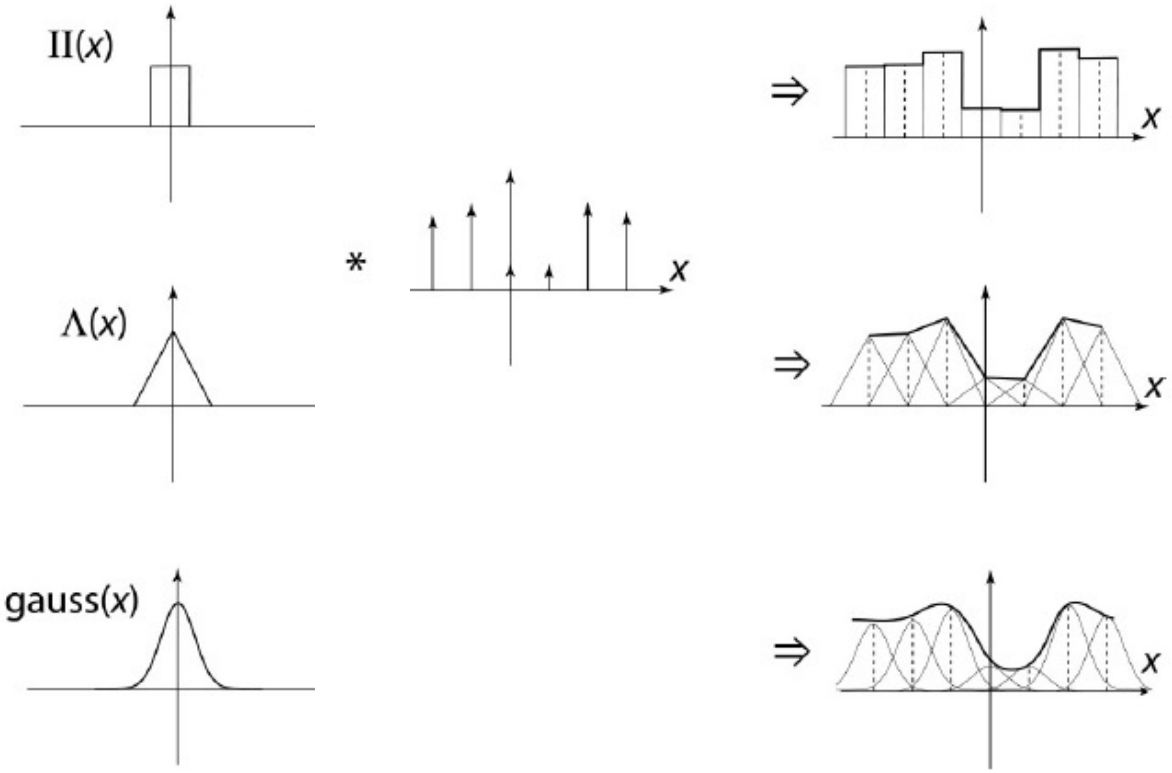
- Dacă nu cunoaştem f ?
  - aproximăm f: $\tilde{f}$
  - folosim filtrarea
  - convertim F în funcția f_F = { F(x), dacă x e întreg; 0, altfel

  - reconstruim f prin convoluție cu un filtru de reconstrucție h

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

Slide adaptat după Steve Seitz

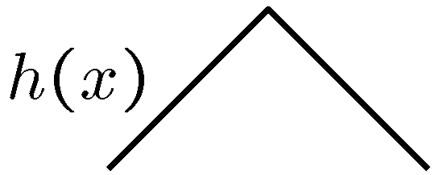
Tipuri de interpolare



Slide adaptat după B. Curless

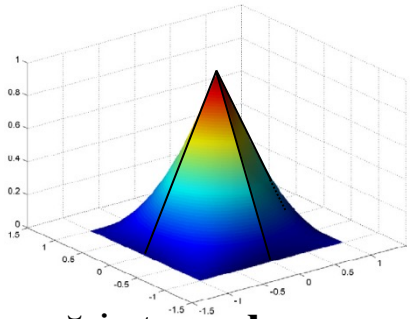
Filtre de reconstrucție

- 1D → 2D



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

$h(x,y)$



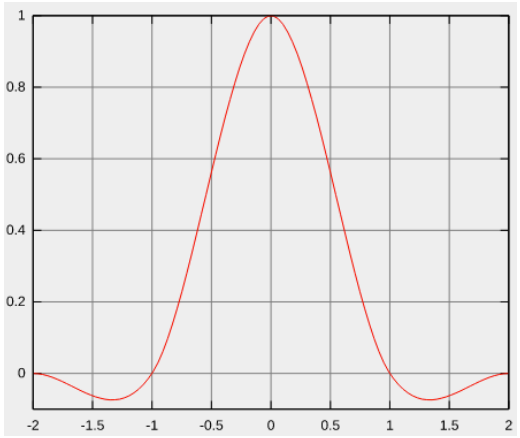
realizează i**nterpolarea biliniară** pe baza celor mai apropiați 4 pixeli

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

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

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

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



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

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

Aplicatie laborator: Redimensionarea imaginilor cu păstrarea conținutului

Redimensionarea imaginilor cu păstrarea conținutului

Imagine inițiala

Redimensionare imagine: vreau să măresc/micșorez lățimea/înălțimea imaginii

Idee: adaug/elimin înșiruiri de pixeli ce conectează extremitățile imaginii ← Cum le aleg?

[Shai & Avidan, SIGGRAPH 2007]

Slide adaptat după K. Grauman

Redimensionarea imaginilor cu păstrarea conținutului

Redimensionare cu păstrarea conținutului

Redimensionare uzuală
(funcția cv.resize în OpenCV)

[Shai & Avidan, SIGGRAPH 2007]

Slide adaptat după K. Grauman

Ideea de bază

https://www.youtube.com/watch?v=6NcIJXTlugc

Slide adaptat după K. Grauman

Ideea de bază

Redimensionare cu păstrarea conținutului

Intuiție:

- Păstrăm conținutul cel mai “interesant”

  → Preferăm să eliminăm/adăugăm pixeli cu gradient mic

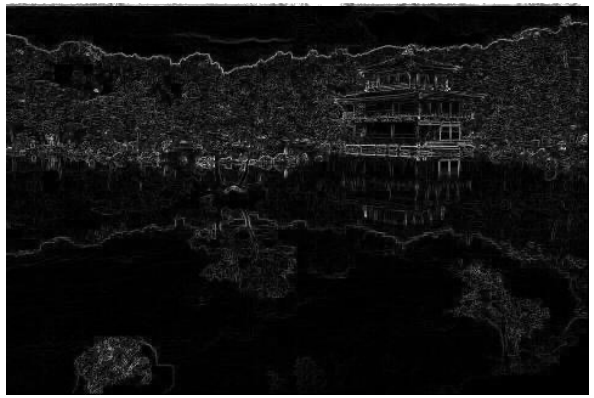
- Pentru a reduce/mări o dimensiune , eliminăm/adăugăm înşiruiri de pixeli (drumuri neregulate) ce conectează extremitățile

  → Soluție optimă folosind programarea dinamică

Slide adaptat după K. Grauman

Ideea de bază – eliminare de drumuri

imaginea I



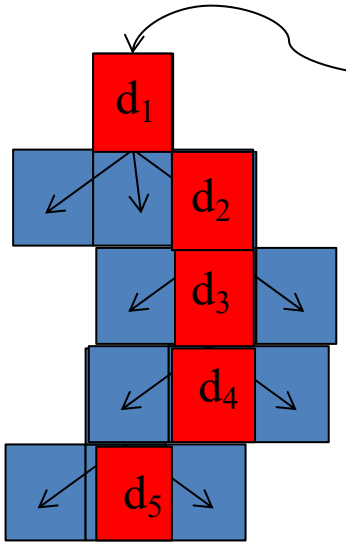
gradientul imaginii I (magnitudine)

$$\nabla I = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}$$

- Eliminăm ‘drumuri’ (înşiruiri de pixeli) fără a afecta vizibil imaginea:
  - măsurăm ‘costul unui drum’ ca suma magnitudinilor gradienților pixelilor ce alcătuiesc drumul
- La fiecare iterație, alegem drumul cu **costul cel mai mic** din imagine

Slide adaptat după K. Grauman

Algoritmul



imaginea I

gradientul imaginii I

$$\nabla I = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}$$

Fie d un drum vertical format din N pixeli: $\mathbf{d} = (d_1, d_2, \ldots, d_N)$

Definim costul unui drum : $Cost(\mathbf{d}) = \sum_{i=1}^{N} \nabla I(d_i)$

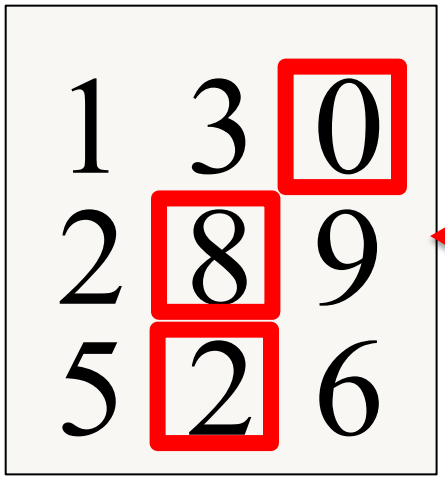
Drumul optim minimizează acest cost: $\mathbf{d}^* = \min_{\mathbf{d}} Cost(\mathbf{d})$

Calculăm drumul optim în mod eficient folosind programarea dinamică

Slide adaptat după K. Grauman

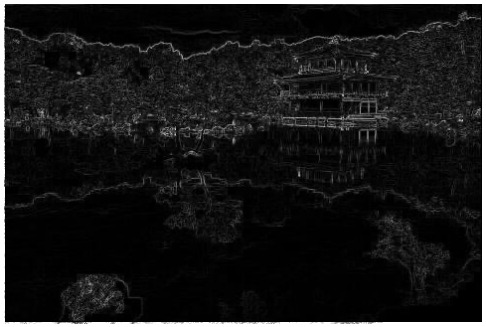
Cum identificăm drumul de cost minim?

- Mai întâi, considerăm o strategie **greedy**:



La fiecare pas aleg cea mai bună soluție locală (aleg pixelii cu gradientul cel mai mic)

Combinarea optimelelor locale NU conduce la optim global



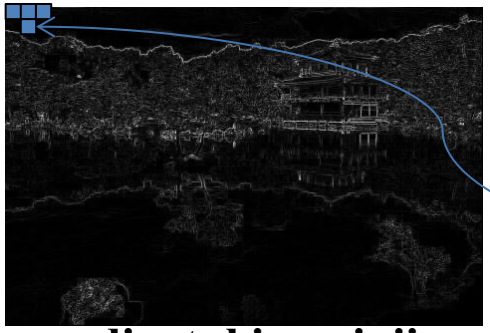
gradientul imaginii

Slide adaptat după K. Grauman

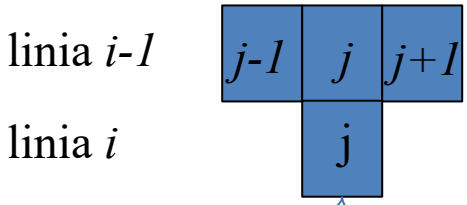
Algoritmul

- Calculăm pentru fiecare pixel (i,j) drumul de cost minim până la (i,j) cu ajutorul celor trei vecini:

$$\mathbf{M}(i,j) = MagGradient(i,j) + \min(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1))$$



gradientul imaginii



Matricea M de costuri ale drumurilor (aici pentru drumuri verticale)

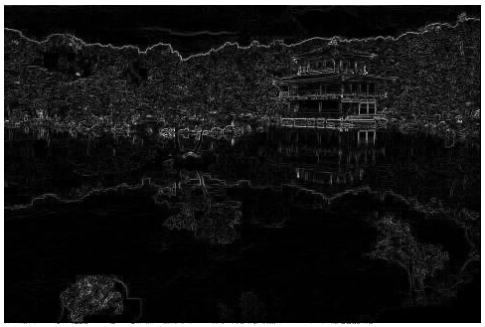
- Valoarea minimă din ultima linie din **M** reprezintă valoarea drumului vertical de cost minim. Poziția costului minim pe ultima linie localizează ultimul pixel din drum.
- Găsim drumul de cost minim mergând înapoi si găsind drumul minim cu ajutrul celor cei 3 vecini de sus din **M**

Slide adaptat după K. Grauman

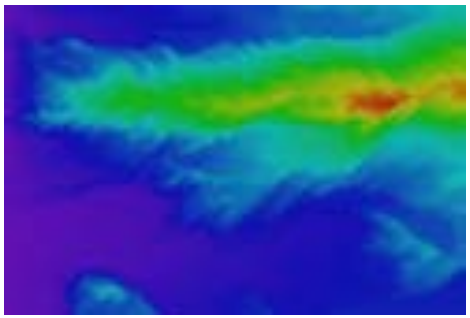
Exemplu – programare dinamică

$$\mathbf{M}(i,j) = MagGradient(i,j) + \min(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1))$$

1	3	0
2	8	9
5	2	6



gradientul imaginii



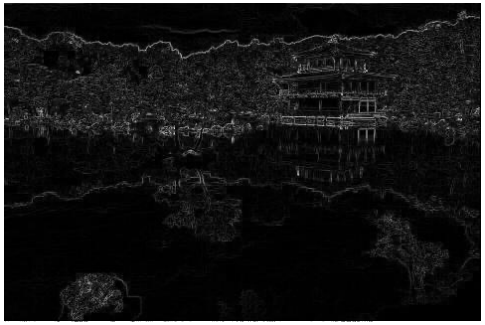
Matricea M de costuri ale drumurilor (aici pentru drumuri verticale)

Exemplu – programare dinamică

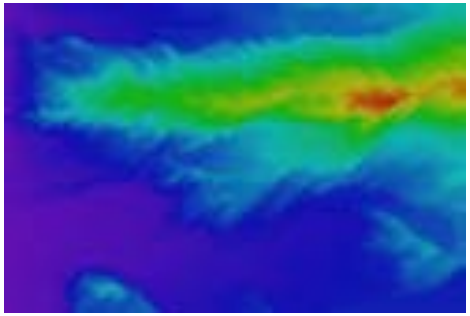
$$\mathbf{M}(i,j) = MagGradient(i,j) + \min(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1))$$

1	3	0
2	8	9
5	2	6

1	3	0
3	8	9
8	5	14



gradientul imaginii

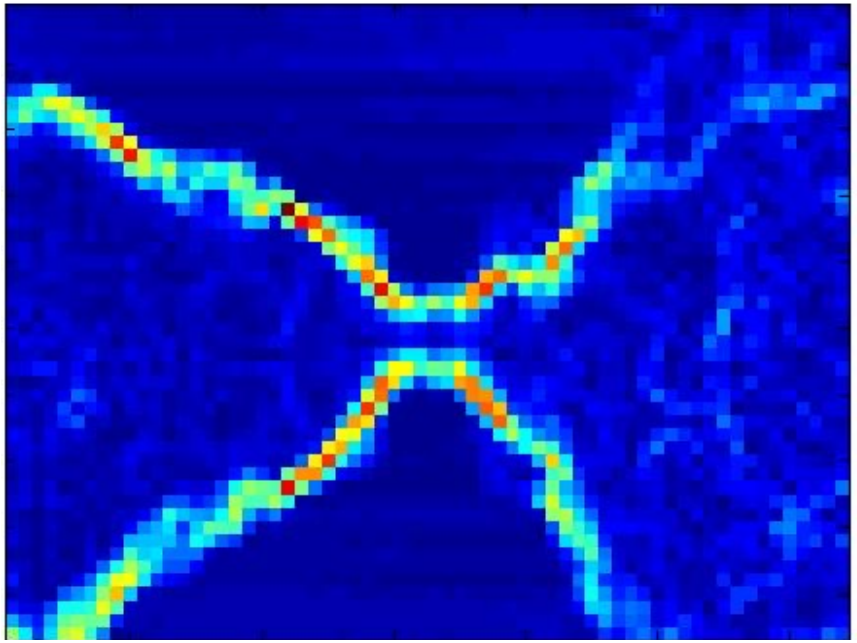


Matricea M de costuri ale drumurilor (aici pentru drumuri verticale)

Exemplu

Imagine inițială

Matricea **M** de costuri ale drumurilor



Albastru = cost mic

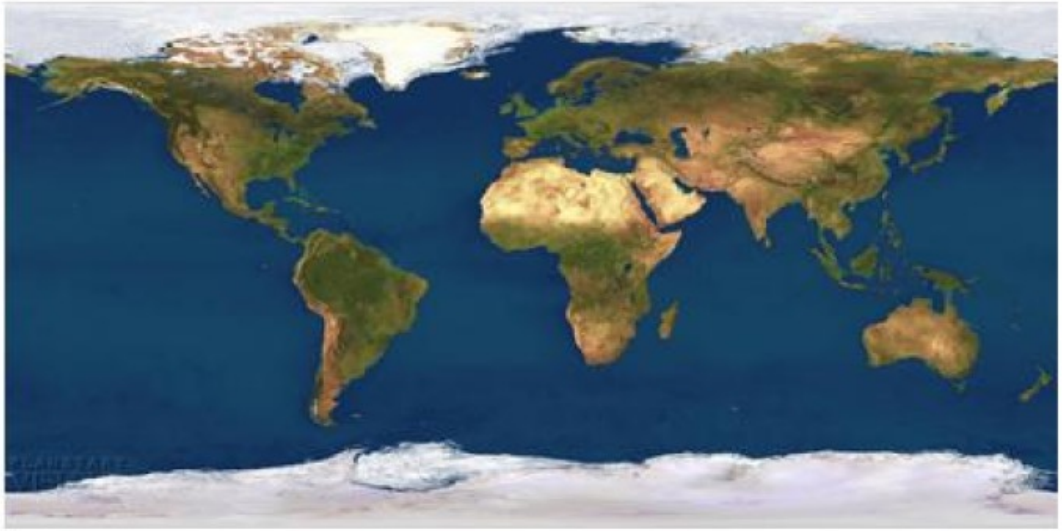
Roşu = cost mare

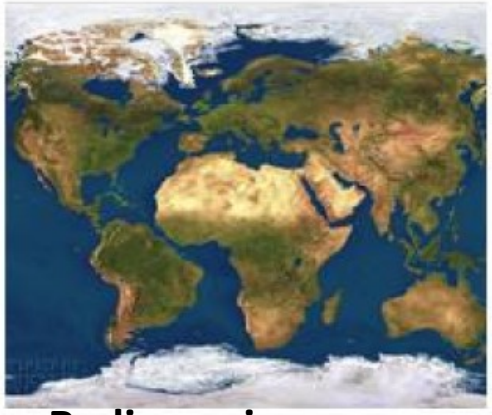
Slide adaptat după K. Grauman

Exemplu

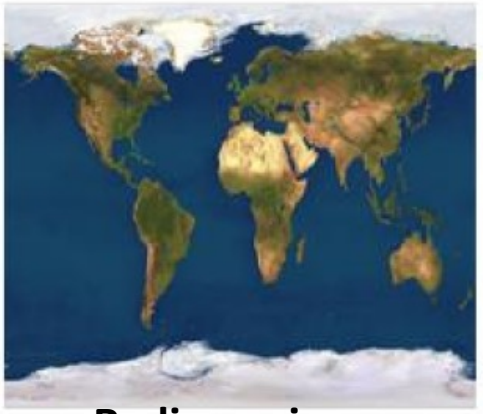
Slide adaptat după K. Grauman

Rezultate





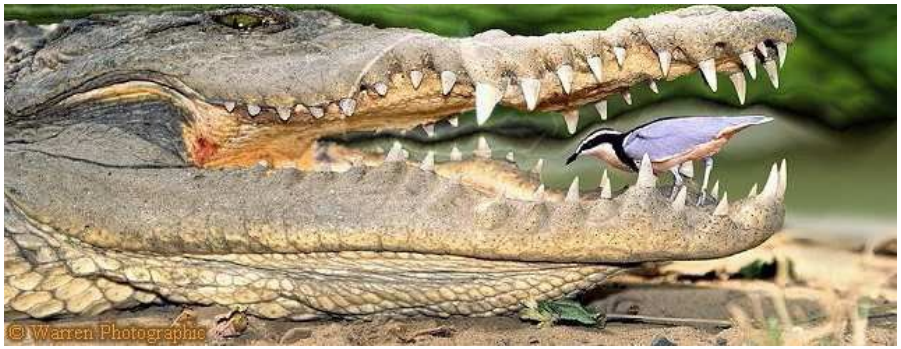
Redimensionare cu păstrarea conținutului



Redimensionare uzuală (imresize)

Slide adaptat după K. Grauman

Rezultate



Redimensionare cu păstrarea conținutului

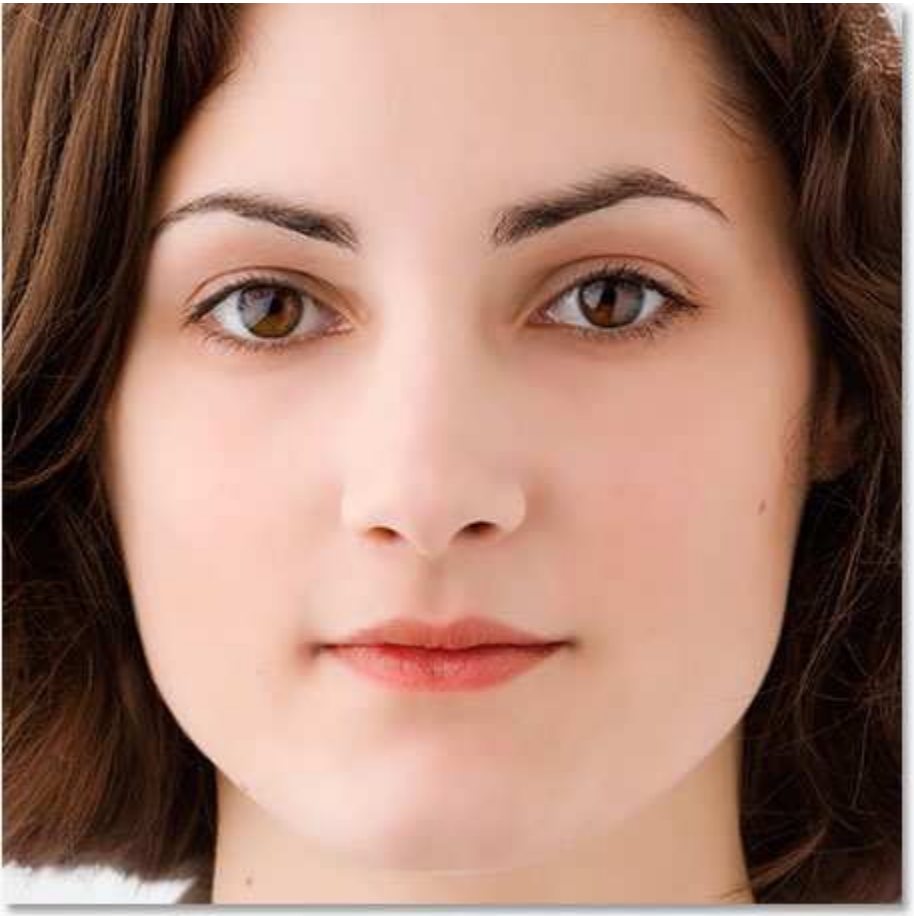
Redimensionare uzuală (imresize)

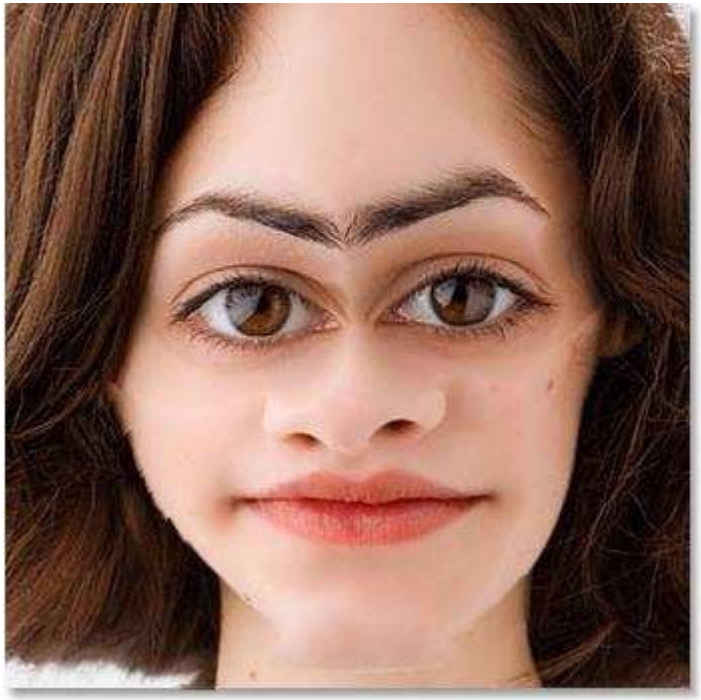
Slide adaptat după K. Grauman

Rezultate - failures

Slide adaptat după K. Grauman

Rezultate - failures





Slide adaptat după K. Grauman

Eliminarea unei regiuni din imagine