Concepte și aplicații în Vederea Artificială

Bogdan Alexe
bogdan.alexe@fmi.unibuc.ro

Cursul 5

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

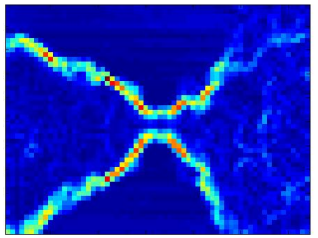
Studenți ce au trimis tema 1

	Nume	Grupa	Observații
1	Caluian Iulian	343	
2	Dranca Constantin		o zi intarziere
3	Octavian Mihai Marcovschi		o zi intarziere
4	Dumitrescu Gabriel Horia	333	o zi intarziere
5	Tomi Andra Cornelia	344	o zi intarziere
6	Botezatu Daniel Andrei		
7	Jitca David	334	
8	Mare Tudor		
9	Isaia Vlad-Lucian	343	
10	Belcineanu Alexandru-Ioan	344	
11	Nedelcu Andreea	344	
12	Alexandru Banu	331	
13	Robert Stancu		
14	Uta Stefana	344	
15	Feraru Andrei Ionut	342	
16	Dragos Giumanca	342	
17	Robu Costin Stefan	332	
18	Teodor Ionescu	331	
19	Banu Robert Emanuel	342	
20	Andrei Zugravu	331	
21	Lupascu Marian	331	
22	Eduard Poesina	334	

Cursul trecut

- Gradienți

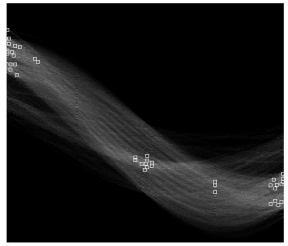

- Tema 2: redimensionarea imaginilor cu păstrarea conținutului

- Cum transformăm gradienții în muchii

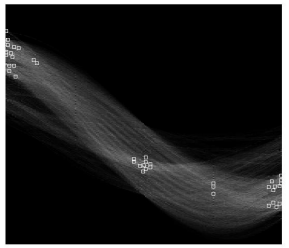


- Aplicație: detectarea liniilor cu transformata Hough

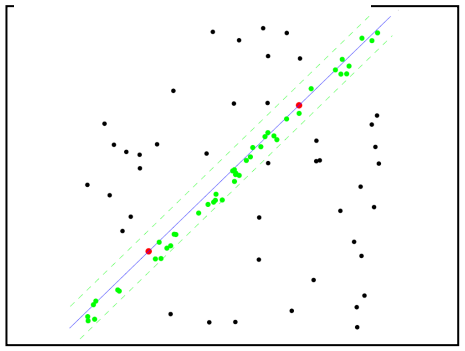


Cursul de azi

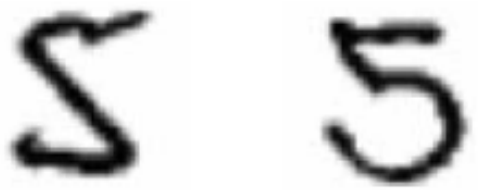
- Aplicație: detectarea liniilor cu transformata Hough



- Aplicație: detectarea liniilor cu RANSAC



- Compararea contururilor

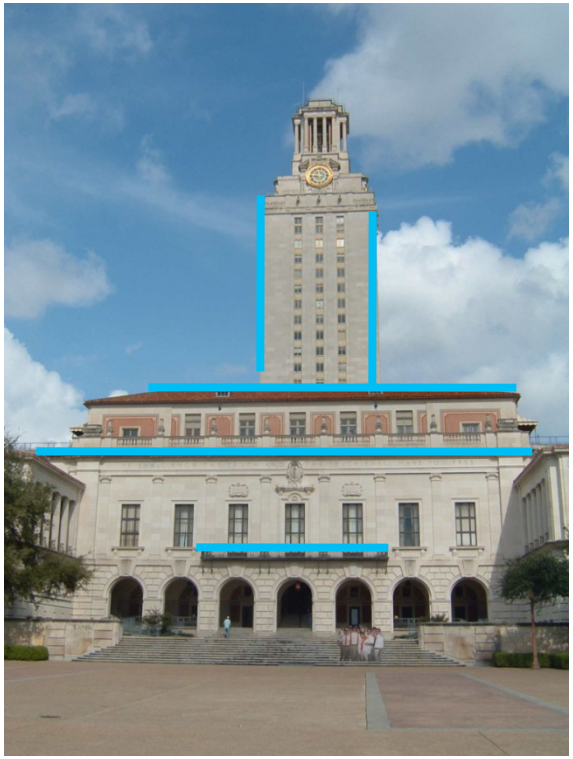


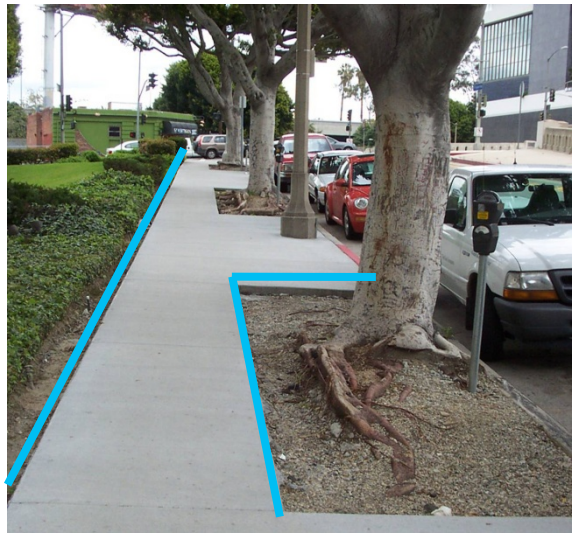
- Textură

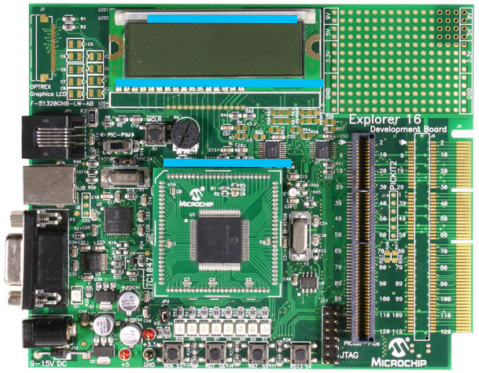
Aplicație: detectarea liniilor cu transformata Hough

Aplicație: detectarea liniilor

- Multe obiecte/suprafețe sunt caracterizate de prezența unor linii drepte





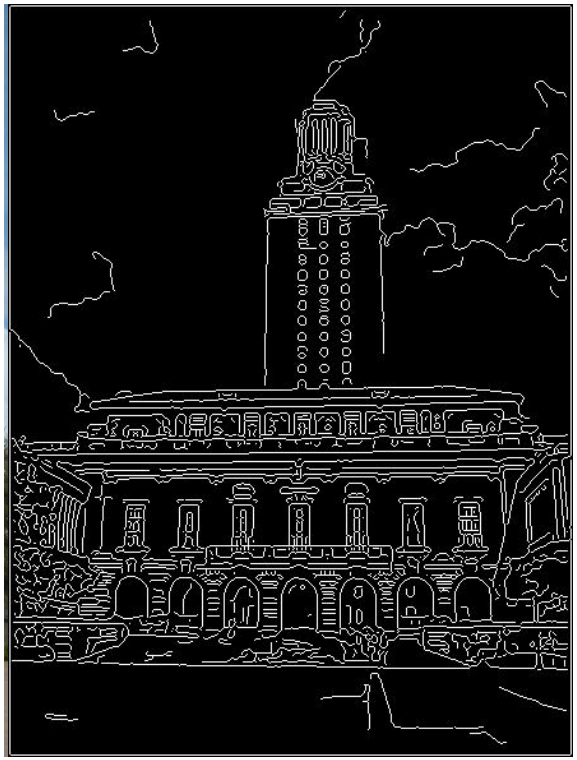


- De ce nu ne rezumăm la a rula un detector de muchii?

Slide adaptat după S. Lazebnik

De ce e dificilă detectarea liniilor

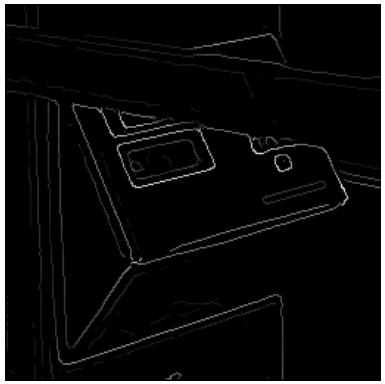
- multe puncte din background considerate ca fiind edgels
- zgomot transformat în edgels

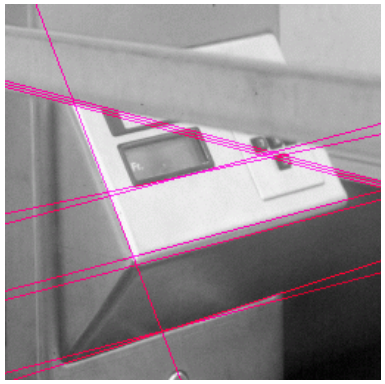


- linii detectate parțial
- cum grupăm punctele în linii?

Slide adaptat după S. Lazebnik

Detectarea liniilor





- date fiind câteva puncte care aparțin unei linii, cum găsim linia?
- câte linii sunt în imagine?
- fiecare puncte cărei linii aparține?

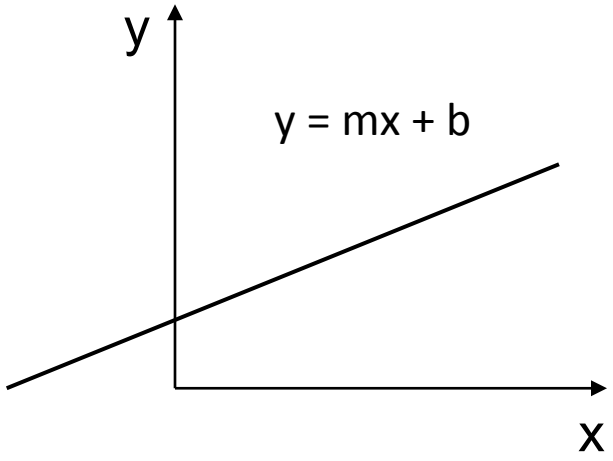
- **Transformata Hough** este o tehnică bazată pe “votare” care răspunde la aceste întrebări.

  Ideea principală:

  1. înregistrează toate liniile posibile care pot conține un punct detectat ca fiind edgel.
  2. găsește linia care primește cele mai multe voturi.

Slide adaptat după S. Lazebnik

Parametrizarea dreptelor



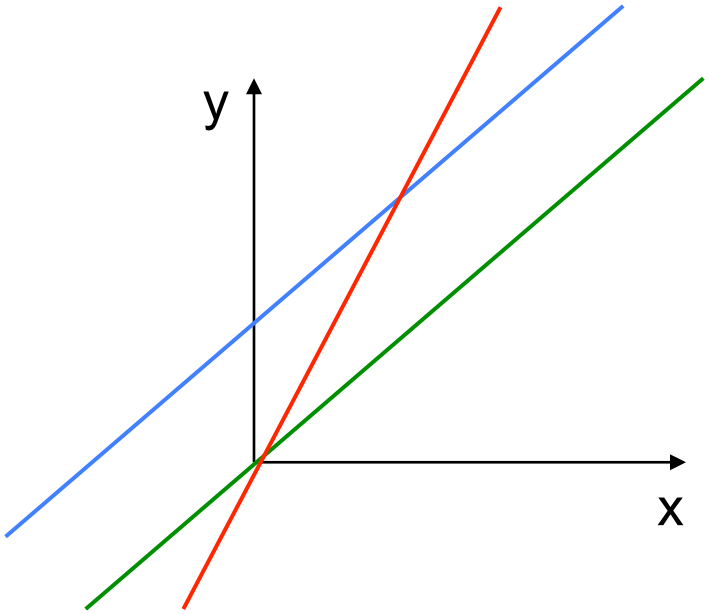
Ecuația unei drepte în planul imaginii: $y = m * x + b$,

m – panta dreptei (definește înclinarea față de axa Ox),

b – deplasarea (definește deplasarea față de originea O(0,0))

(m,b) sunt parametri dreptei

Parametrizarea dreptelor. Exemple



d_1: y = x, adică m=1, b = 0

d_2: y = x + 2, adică m=1, b = 2

d_3: y = 2x, adică m=2, b = 0

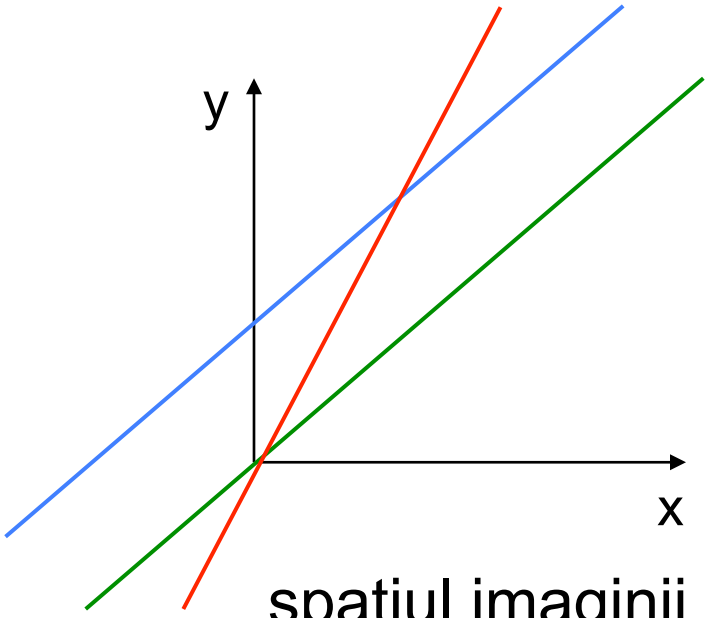
Ecuația unei drepte în planul imaginii: $y = m * x + b$,

m – panta dreptei (definește înclinarea față de axa Ox),

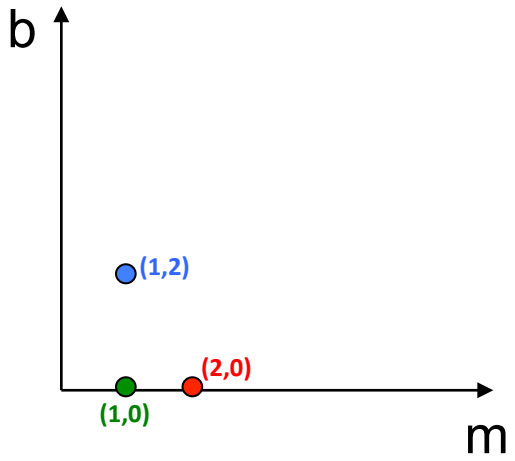
b – deplasarea (definește deplasarea față de originea O(0,0))

(m,b) sunt parametri dreptei

Spațiul Hough al parametrilor



spațiul imaginii



spațiul Hough
(al parametrilor)

d_1: y = x, adică m=1, b = 0

d_2: y = x + 2, adică m=1, b = 2

d_3: y = 2x, adică m=2, b = 0

d_1: m=1, b = 0 -> (1,0)

d_2: m=1, b = 2 -> (1,2)

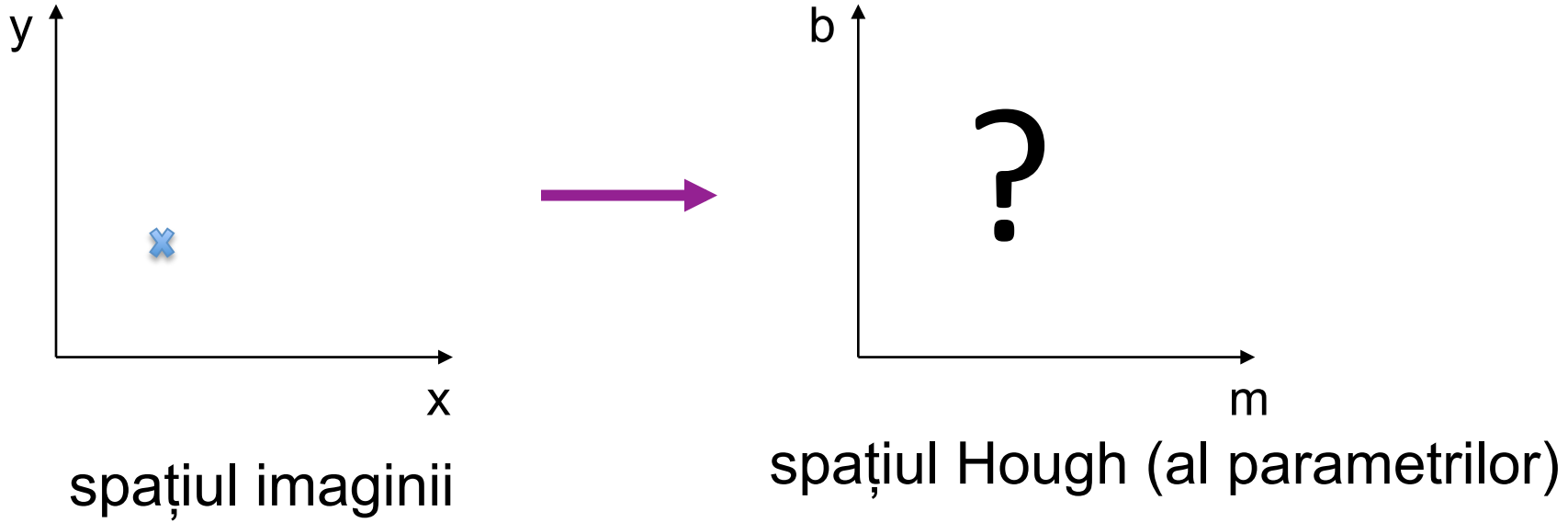
d_3: m=2, b = 0 -> (2,0)

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Corespondența dintre cele două spații

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

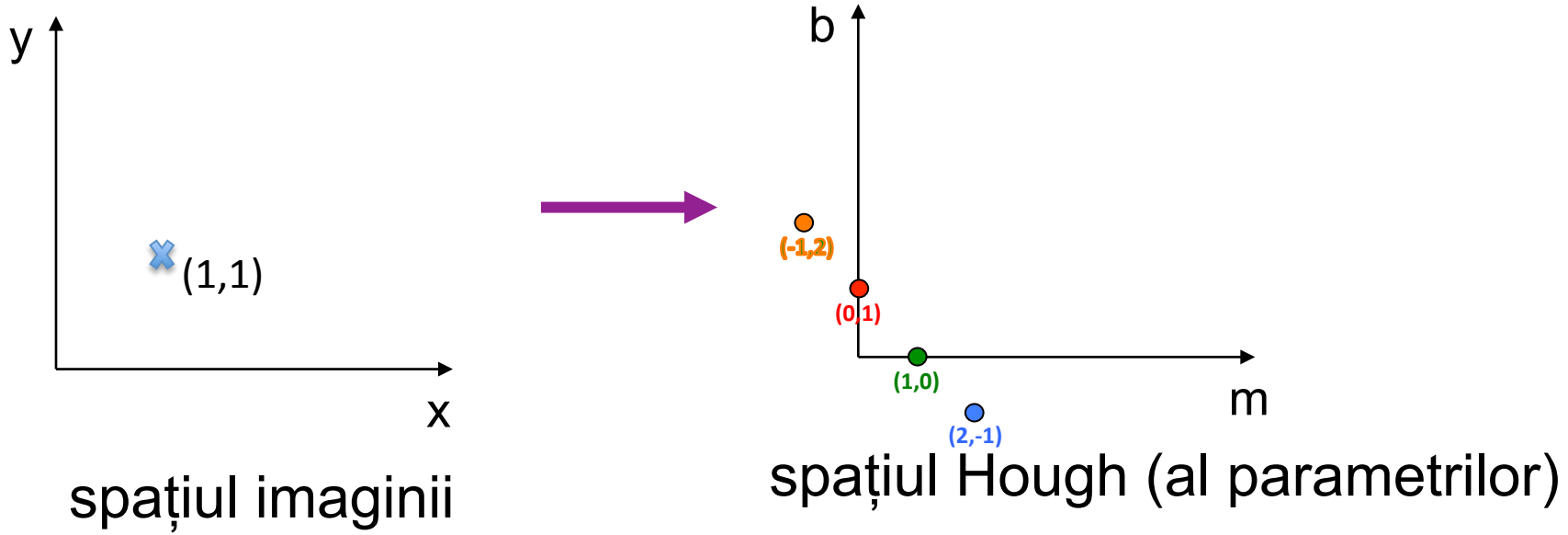
Se dă un punct (x_0,y_0) în spațiul imaginii. Ce îi corespunde în spațiul Hough?



Corespondența dintre cele două spații

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Se dă un punct (x_0,y_0) în spațiul imaginii. Ce îi corespunde în spațiul Hough?



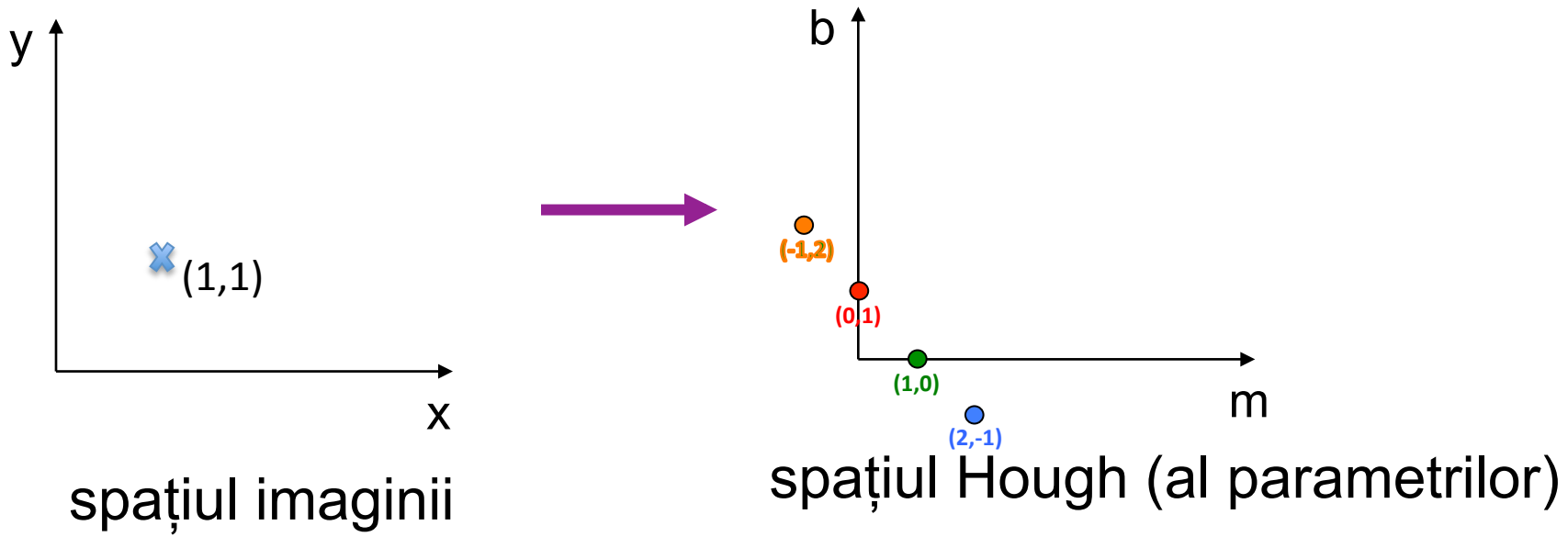
Prin punctul (1,1) trec o infinitate de drepte de forma y = mx+b:

y = x (m=1, b=0); y = 2x-1 (m=2,b=-1), y = 1 (m=0, b = -1), y = -x + 2 (m=-1, b =2),

Corespondența dintre cele două spații

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Un punct (x_0,y_0) în spațiul imaginii corespunde unei drepte în spațiul Hough.



Prin punctul (1,1) trec o infinitate de drepte de forma y = mx+b:

y = x (m=1, b=0); y = 2x-1 (m=2,b=-1), y = 1 (m=0, b = -1), y = -x + 2 (m=-1, b =2),

Toate cele 4 puncte (1,0), (2,-1), (0,-1), (-1,2) stau pe dreapta b = -m+1

Corespondența dintre cele două spații

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Un punct (x_0,y_0) în spațiul imaginii corespunde unei drepte în spațiul Hough.

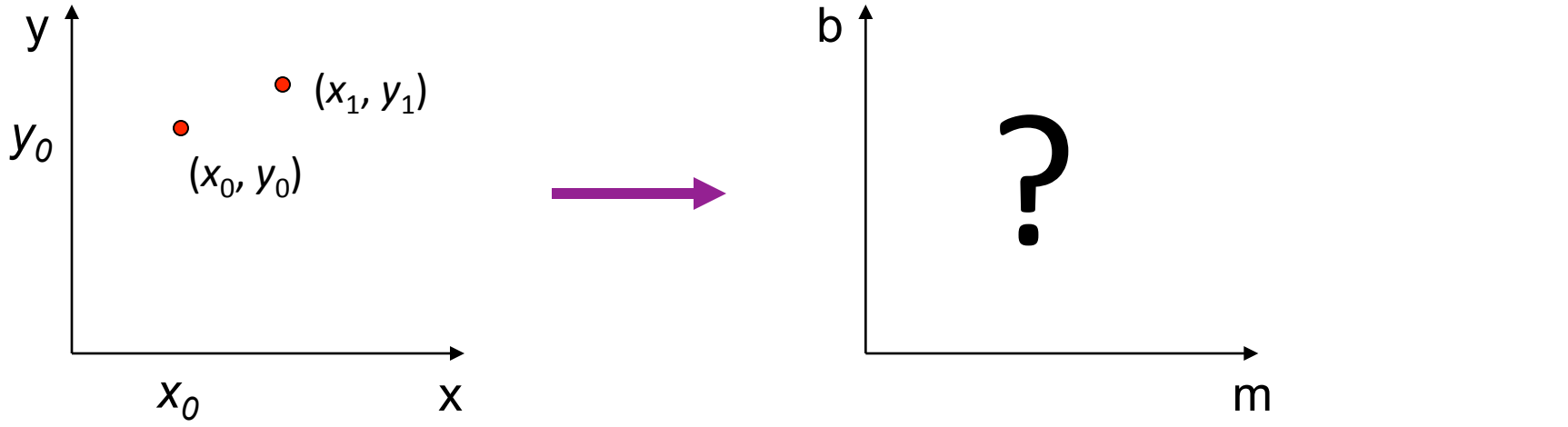
Ecuația unei drepte în planul imaginii: $y = m * x + b$,

Dreptele care trec prin punctul (x_0,y_0) pot avea orice pantă m și orice deplasare b cu condiția ca $y_0 = m * x_0 + b$.

$y_0 = m * x_0 + b \Leftrightarrow b = -x_0 * m + y_0$ (dreapta cu panta $-x_0$ și deplasare y_0)

(x_0,y_0) = (1,1) => b = -m+1

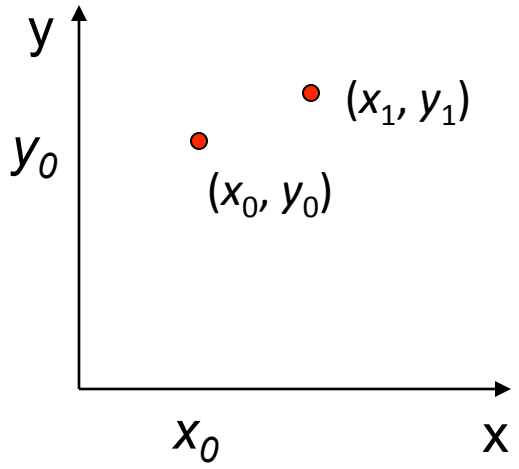
Găsirea liniilor într-o imagine folosing spațiul Hough



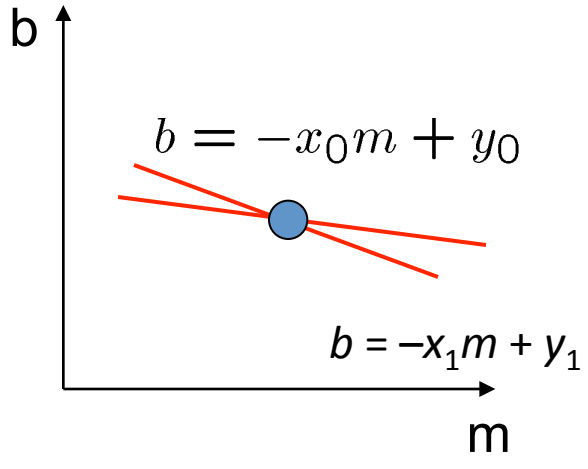
Care sunt parametri liniei care conține punctele (x_0, y_0) și (x_1, y_1)?

Slide adaptat după S. Lazebnik

Găsirea liniilor într-o imagine folosing spațiul Hough



spațiul imaginii



spațiul Hough (al parametrilor)

Care sunt parametri liniei care conține punctele (x_0, y_0) și (x_1, y_1)?

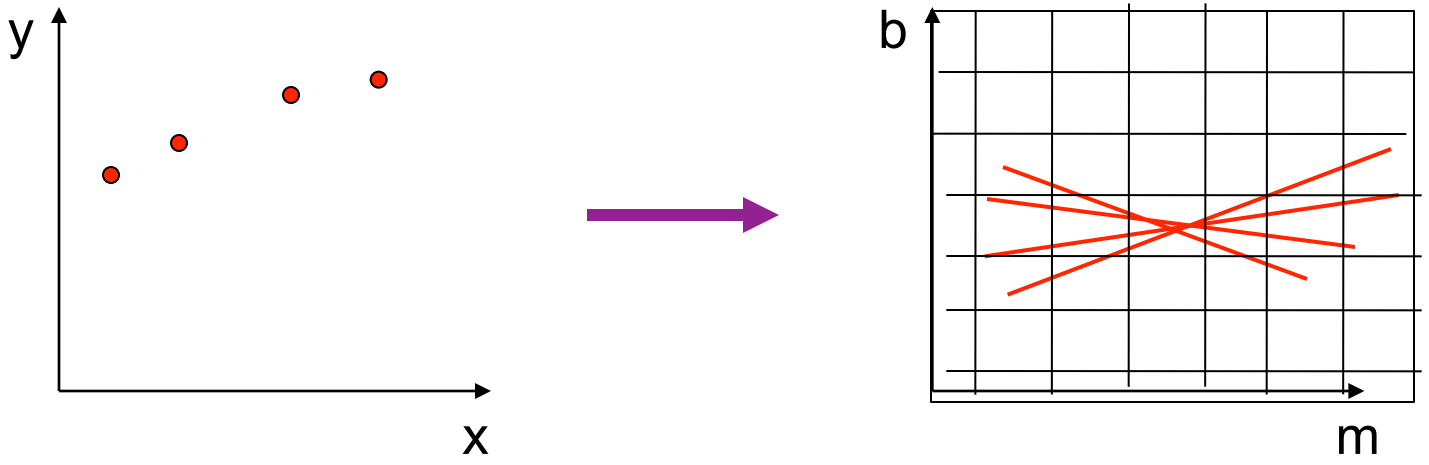
Punctului (x_0,y_0) îi corespunde dreapta de ecuație $b = -x_0m + y_0$

Punctului (x_1,y_1) îi corespunde dreapta de ecuație $b = -x_1m + y_1$

Dreptei d care conține punctele (x_0,y_0), (x_1,y_1) îi corespunde în spațiul Hough un punct. Acest punct se obține ca fiind intersecția dreptelor de ecuație $b = -x_0m + y_0$ și $b = -x_1m + y_1$

Slide adaptat după S. Lazebnik

Găsirea liniilor într-o imagine: algoritmul Hough



spațiul imaginii

spațiul Hough
(al parametrilor)

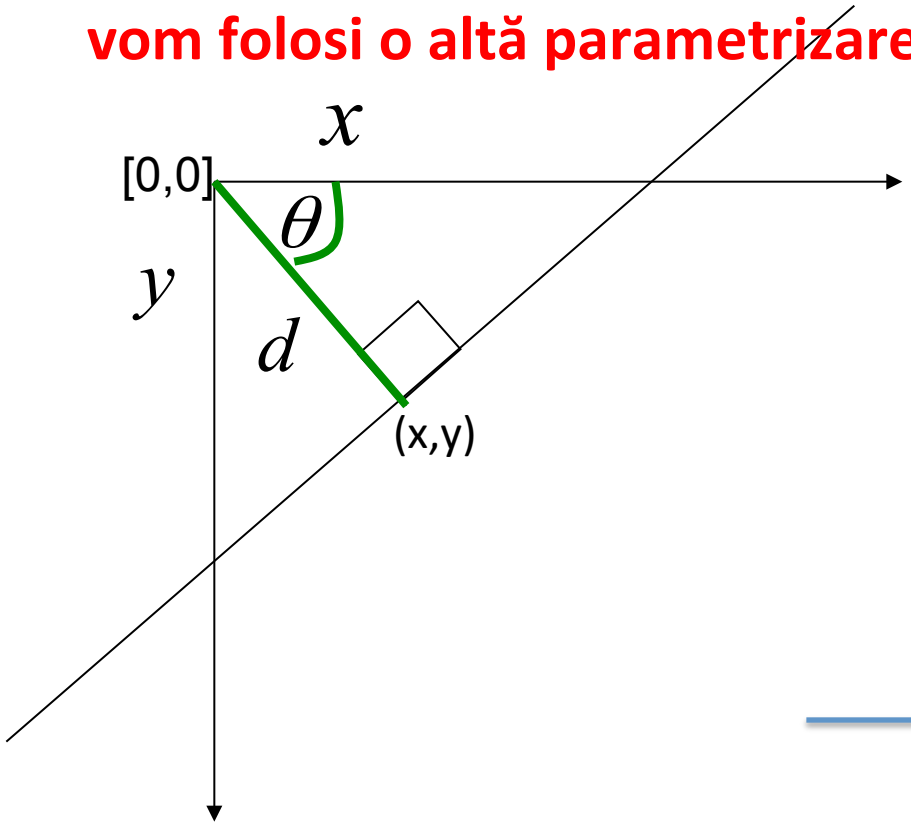
Cum putem folosi observația anterioară pentru a găsi parametri (m,b) ce definesc linia cea mai probabilă în spațiul imaginii?

- fiecare punct detectat ca fiind edgel în spațiul imaginii va vota pentru o mulțime de parametri în spațiul Hough
- acumulăm voturi în intervale discrete; parametri cu cel mai mare număr de voturi determină linia din spațul imaginii

Slide adaptat după S. Lazebnik

Reprezentarea în coordonate polare pentru detectarea liniilor

Probleme cu spațiul Hough al parametrilor (*m,b*): m poate lua o valoare infinită, probleme pentru linii verticale. **Din acest motiv vom folosi o altă parametrizare (coordonate polare):**



d : distanța perpendicularei din origine la linie

θ : unghiul dintre perpendiculară și axa Ox

$$d\cos\theta = x$$

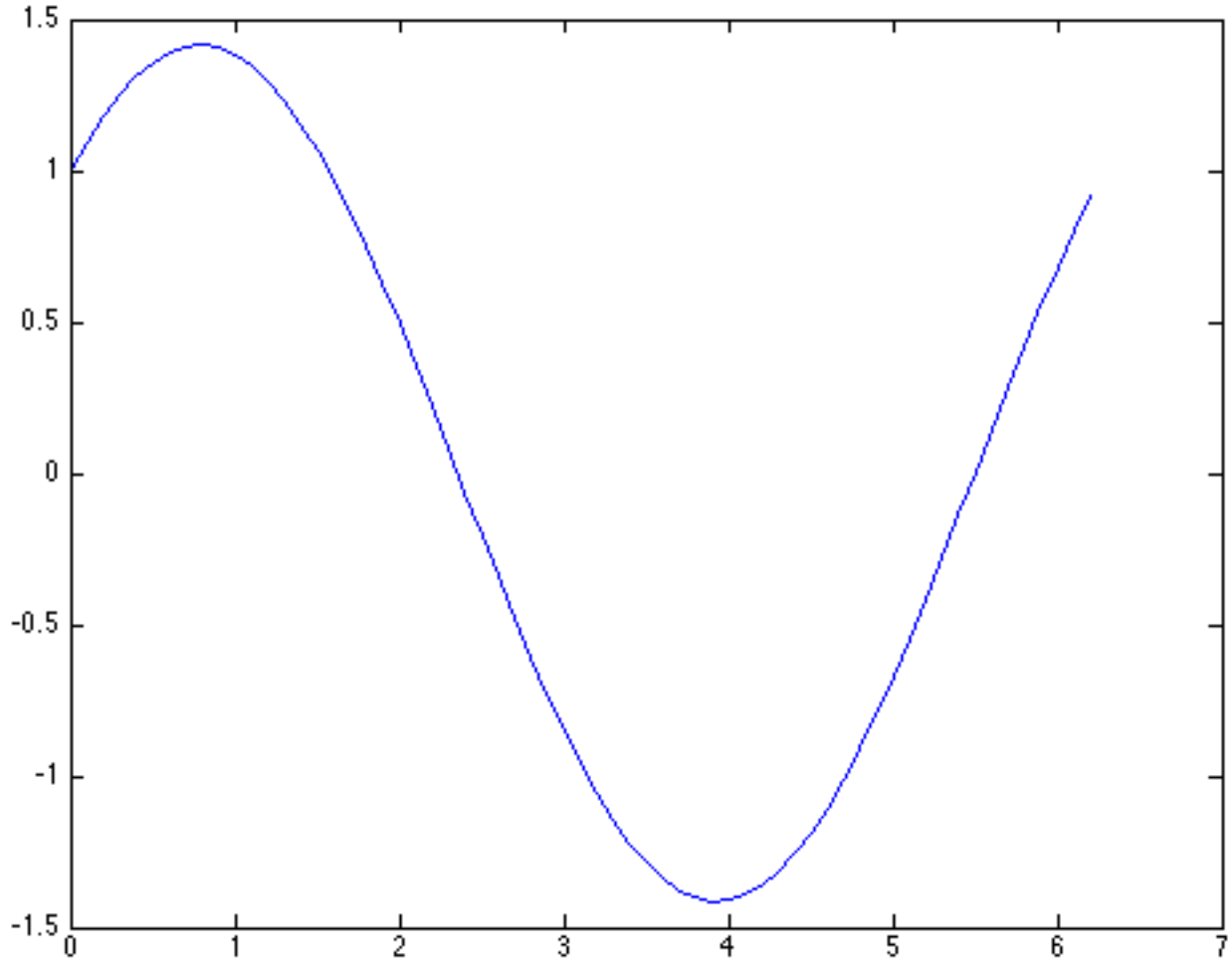
$$d\sin\theta = y$$

$$x\cos\theta + y\sin\theta = d$$

Punct în spațiul imaginii→ curbă sinusoidală în spațiul Hough

Slide adaptat după S. Lazebnik

```
>> x = 1; y =1;
theta = 0:0.1:2*pi
d = x*cos(theta) + y*sin(theta)
figure, plot(theta,d)
```

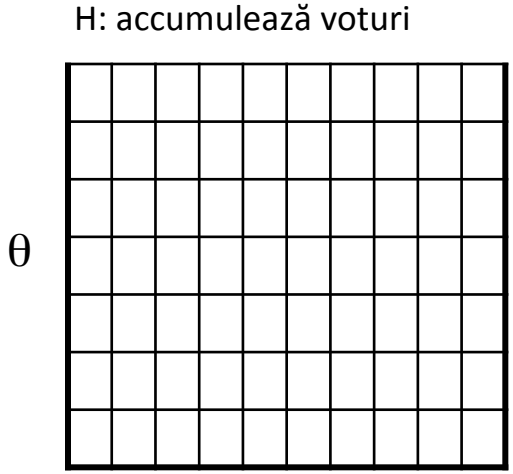


Algoritmul transformatei Hough

- folosește coordonate polare

$$x\cos\theta + y\sin\theta = d$$

H: accumulează voturi



Algoritmul transformatei Hough

1. inițializează H[d, θ]=0
2. pentru fiecare punct edgel din imagine I[x,y]

   pentru θ = 0 to 180 // [0, 2*pi] e împărțit în intervale

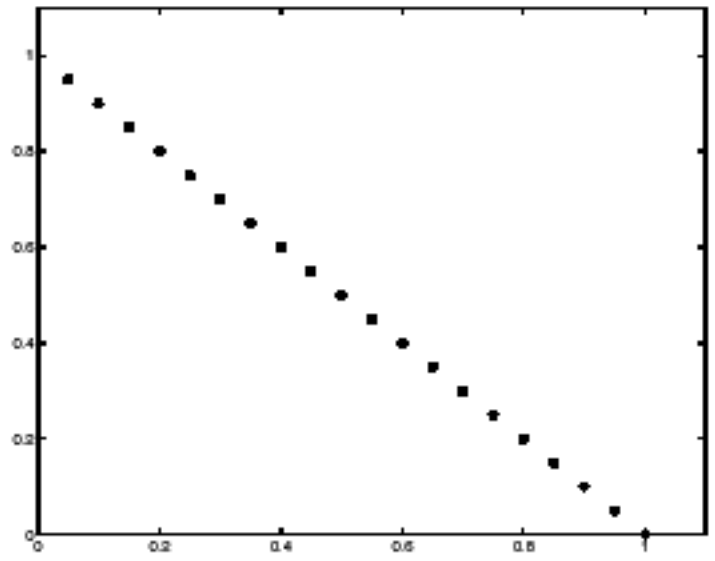
   $d = x\cos\theta + y\sin\theta$

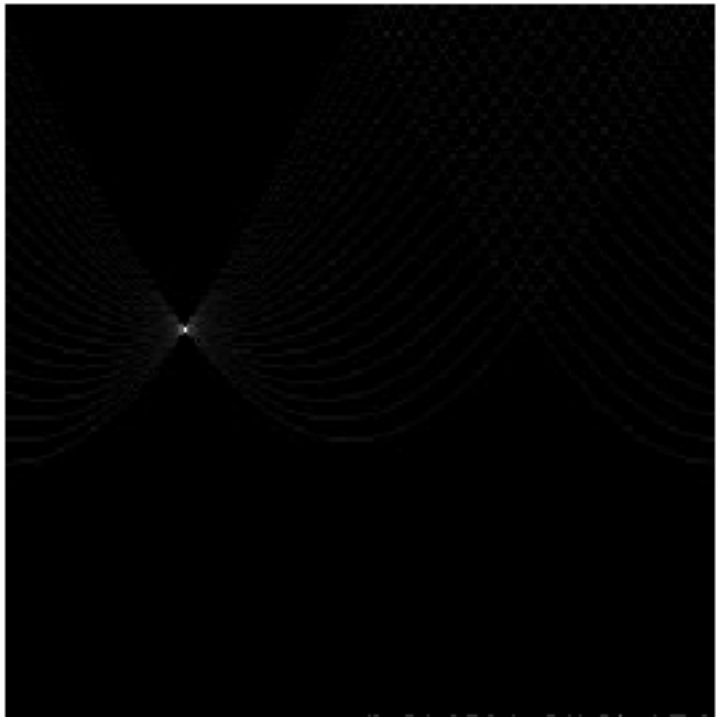
   H[d, θ] += 1
3. găsește valorile (d, θ) pentru care H[d, θ] este maxim
4. linia detectată în imagine este dată de: $d = x\cos\theta + y\sin\theta$

DEMO: http://www.dis.uniroma1.it/~iocchi/slides/icra2001/java/hough.html

Slide adaptat după S. Lazebnik

Exemplu: transformata Hough pentru linii





spațiul imaginii
coordonatele punctelor edgels

Voturi

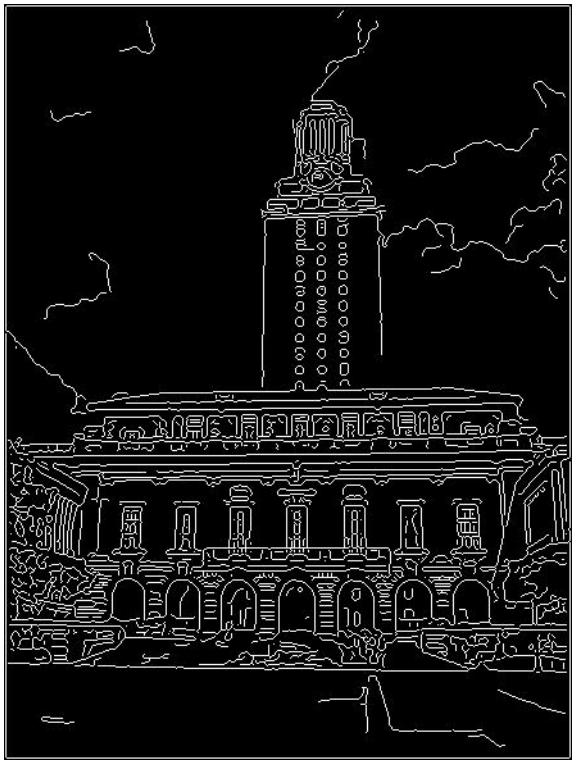
culoare deschisă = # mare de voturi
negru = zero voturi

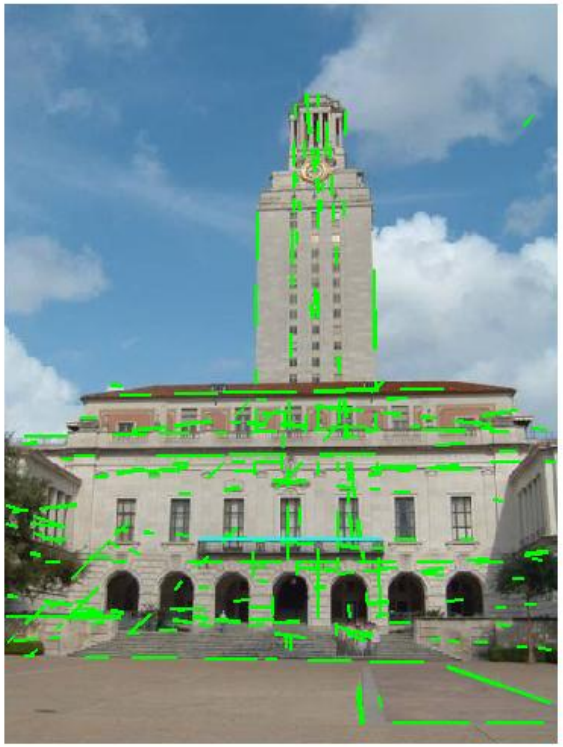
Slide adaptat după S. Lazebnik

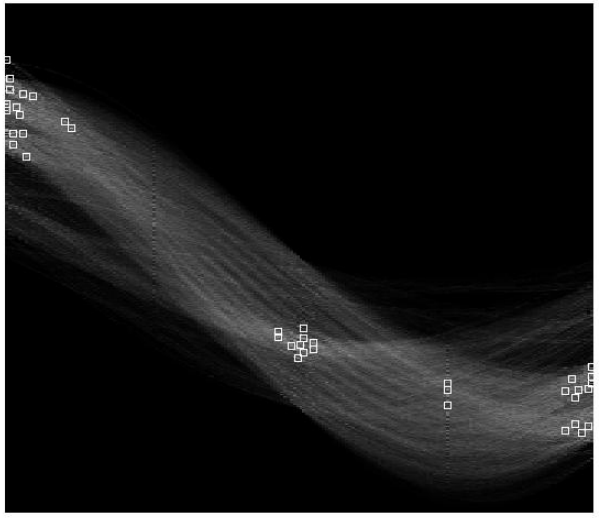
Exemplu: transformata Hough pentru linii

Pătrat:

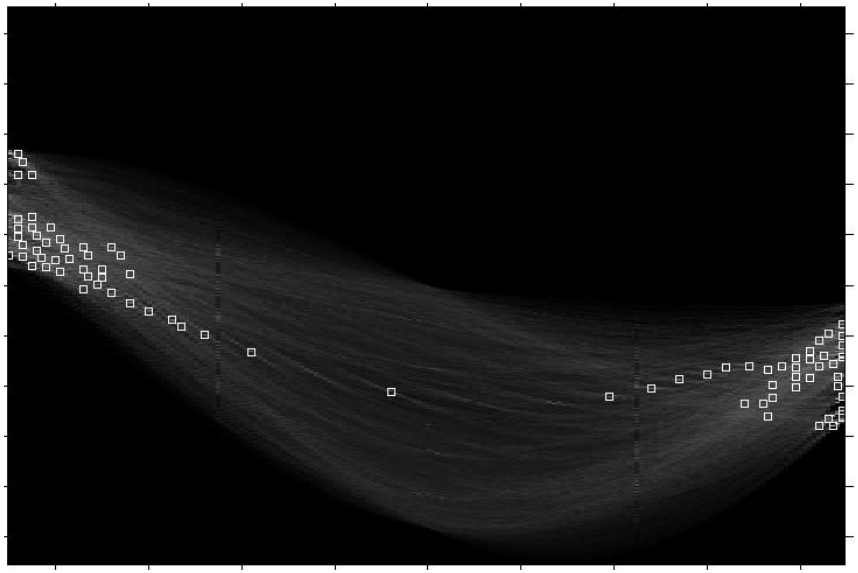
Slide adaptat după S. Lazebnik

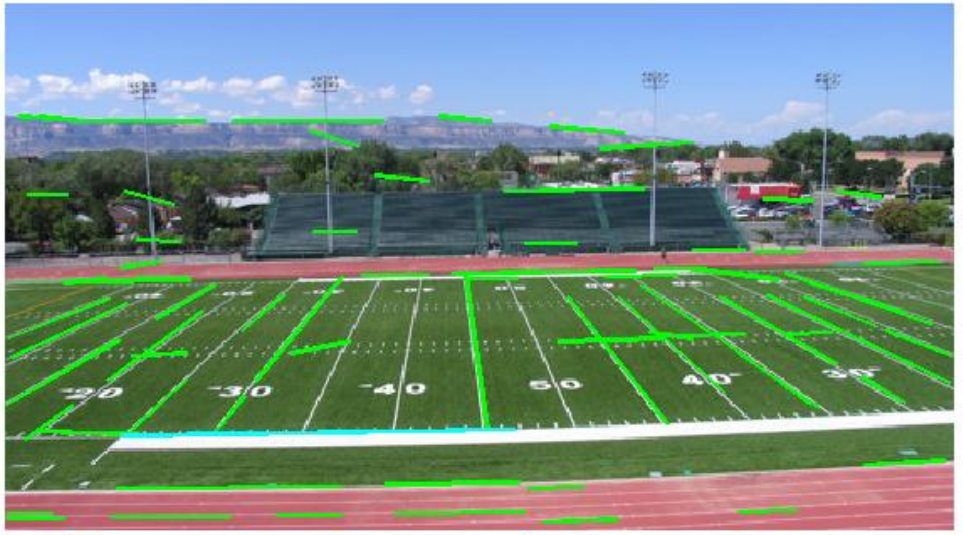






Slide adaptat după S. Lazebnik

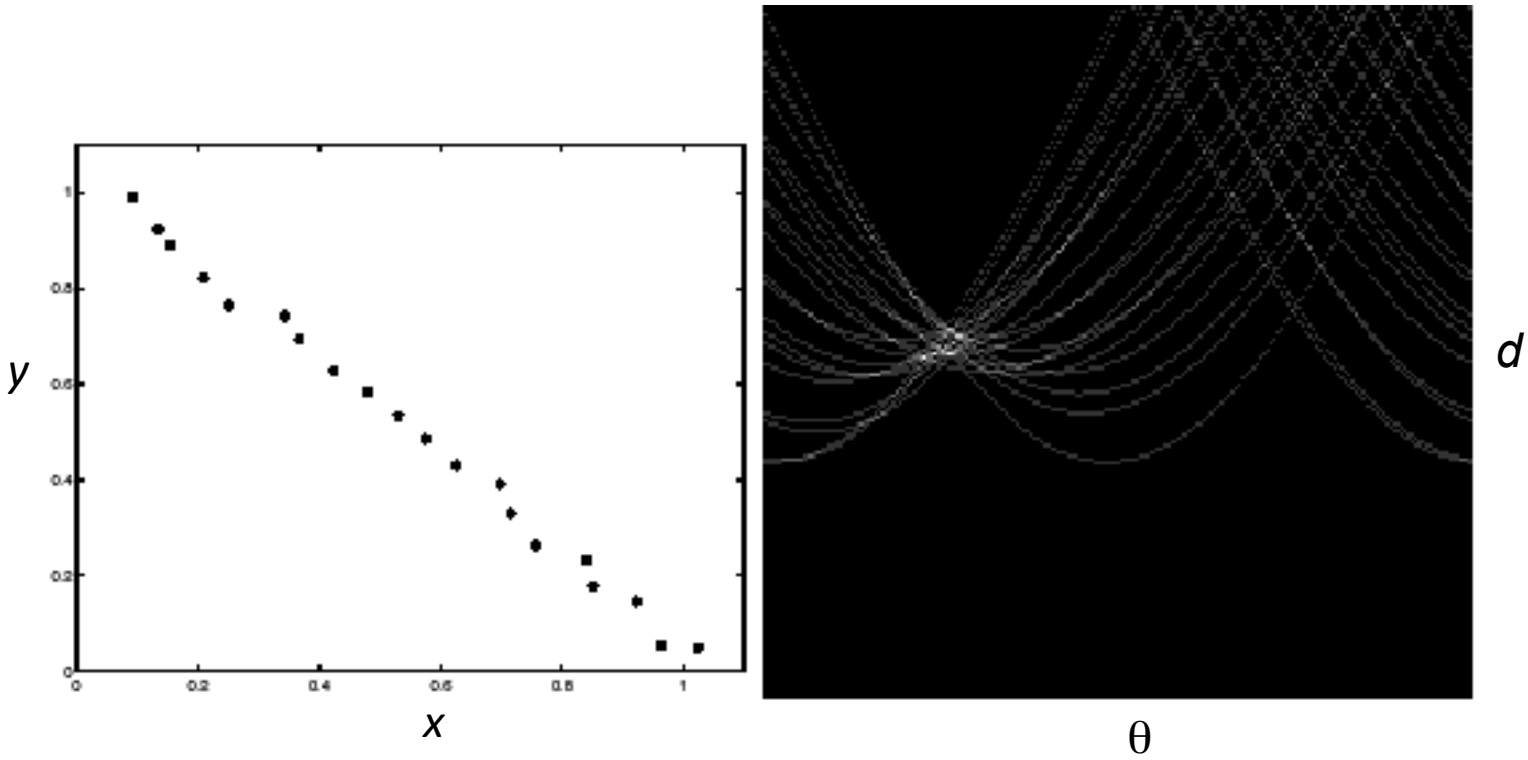




Sunt afișate cele mai lungi segmente

Slide adaptat după S. Lazebnik

Impactul zgomotului asupra transformatei Hough



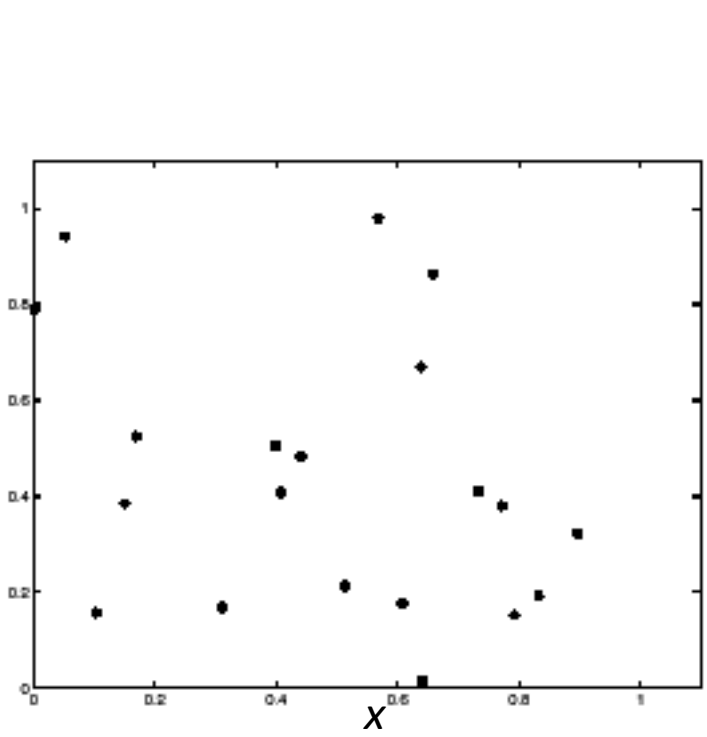
spațiul imaginii
coordonatele punctelor edgels

Voturi

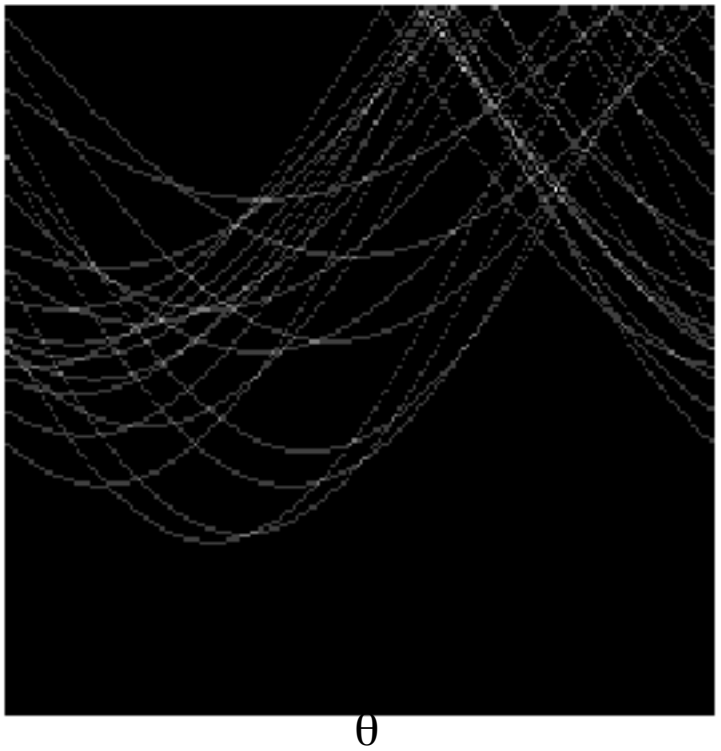
culoare deschisă = # mare de voturi
negru = zero voturi

Slide adaptat după S. Lazebnik

Impactul zgomotului asupra transformatei Hough



spațiul imaginii
coordonatele punctelor edgels

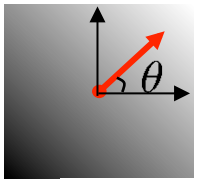


Voturi

Totul pare a fi zgomot sau puncte edgels aleatoare. Se observă niște peak-uri în spațiul parametrilor.

Slide adaptat după S. Lazebnik

Extensii



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Extensia 1: folosește gradientul imaginii

1. inițializează H[d, θ]=0 (la fel)
2. pentru fiecare punct edgel din imagine I[x,y]

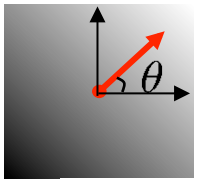
   θ = gradientul imaginii la (x,y)

   $d = x\cos\theta + y\sin\theta$

   H[d, θ] += 1
3. găsește valorile (d, θ) pentru care H[d, θ] este maxim (la fel)
4. linia detectată în imagine este dată de: $d = x\cos\theta + y\sin\theta$

Slide adaptat după S. Lazebnik

Extensii



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Extensia 1: folosește gradientul imaginii

1. inițializează H[d, θ]=0 (la fel)
2. pentru fiecare punct edgel din imagine I[x,y]

   θ = gradientul imaginii la (x,y)

   $d = x\cos\theta + y\sin\theta$

   H[d, θ] += 1
3. găsește valorile (d, θ) pentru care H[d, θ] este maxim (la fel)
4. linia detectată în imagine este dată de: $d = x\cos\theta + y\sin\theta$

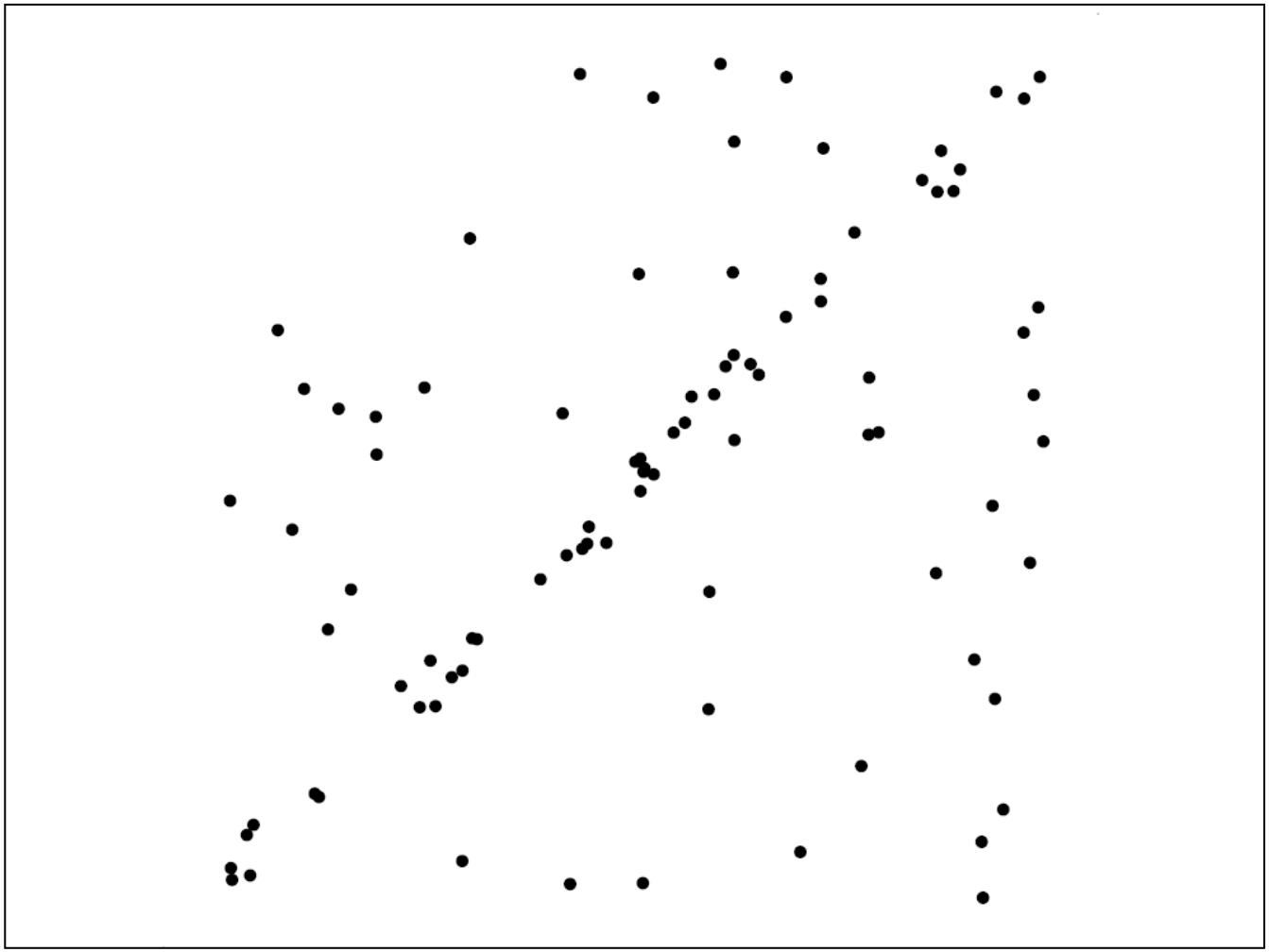
Extensia 2: H[d, θ] += magnitudine gradient (x,y)

Slide adaptat după S. Lazebnik

Aplicație: detectarea liniilor cu algoritmul RANSAC

RANSAC pentru detectarea liniilor

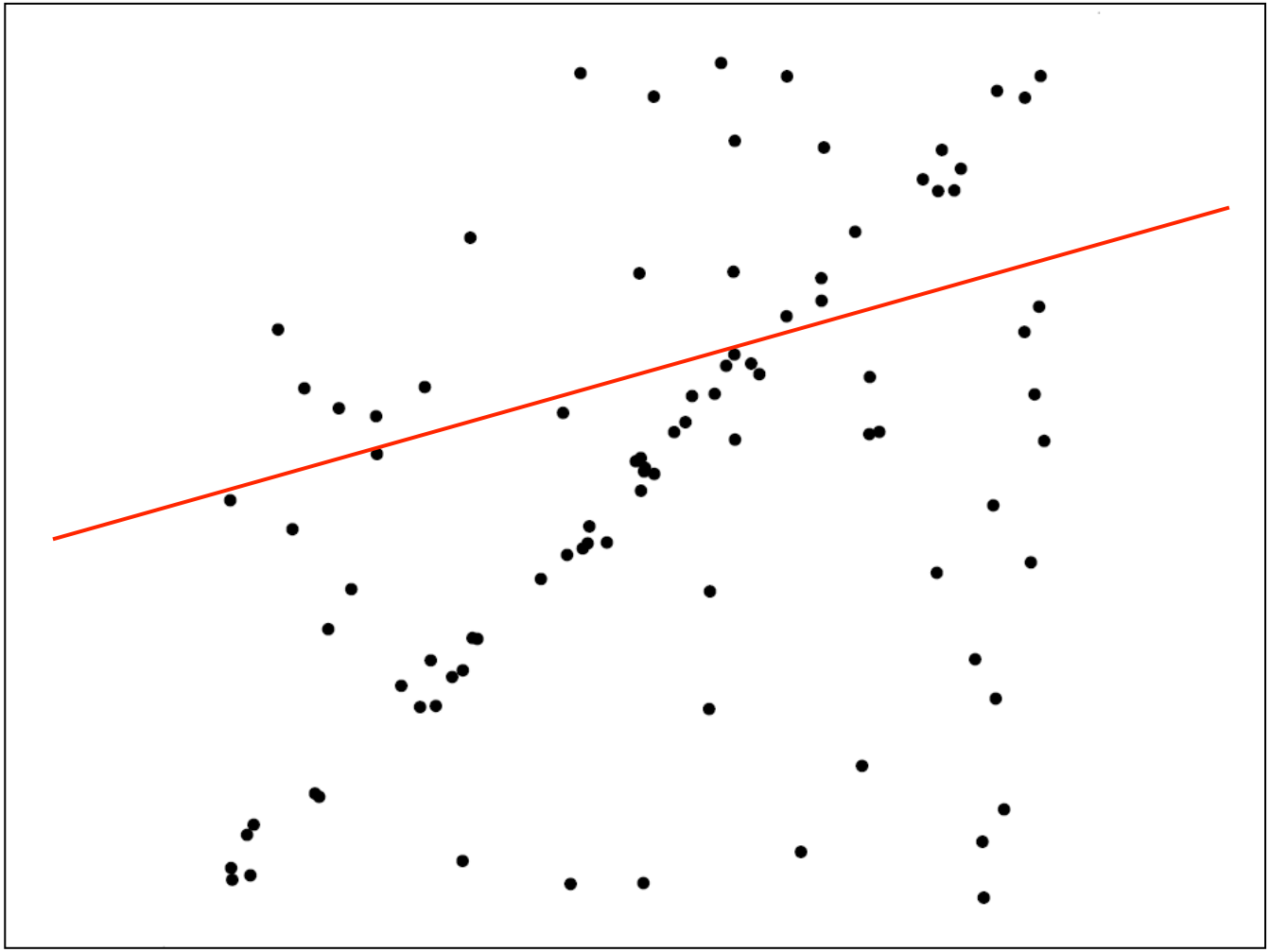
Unde se află linia?



Slide adaptat după S. Lazebnik

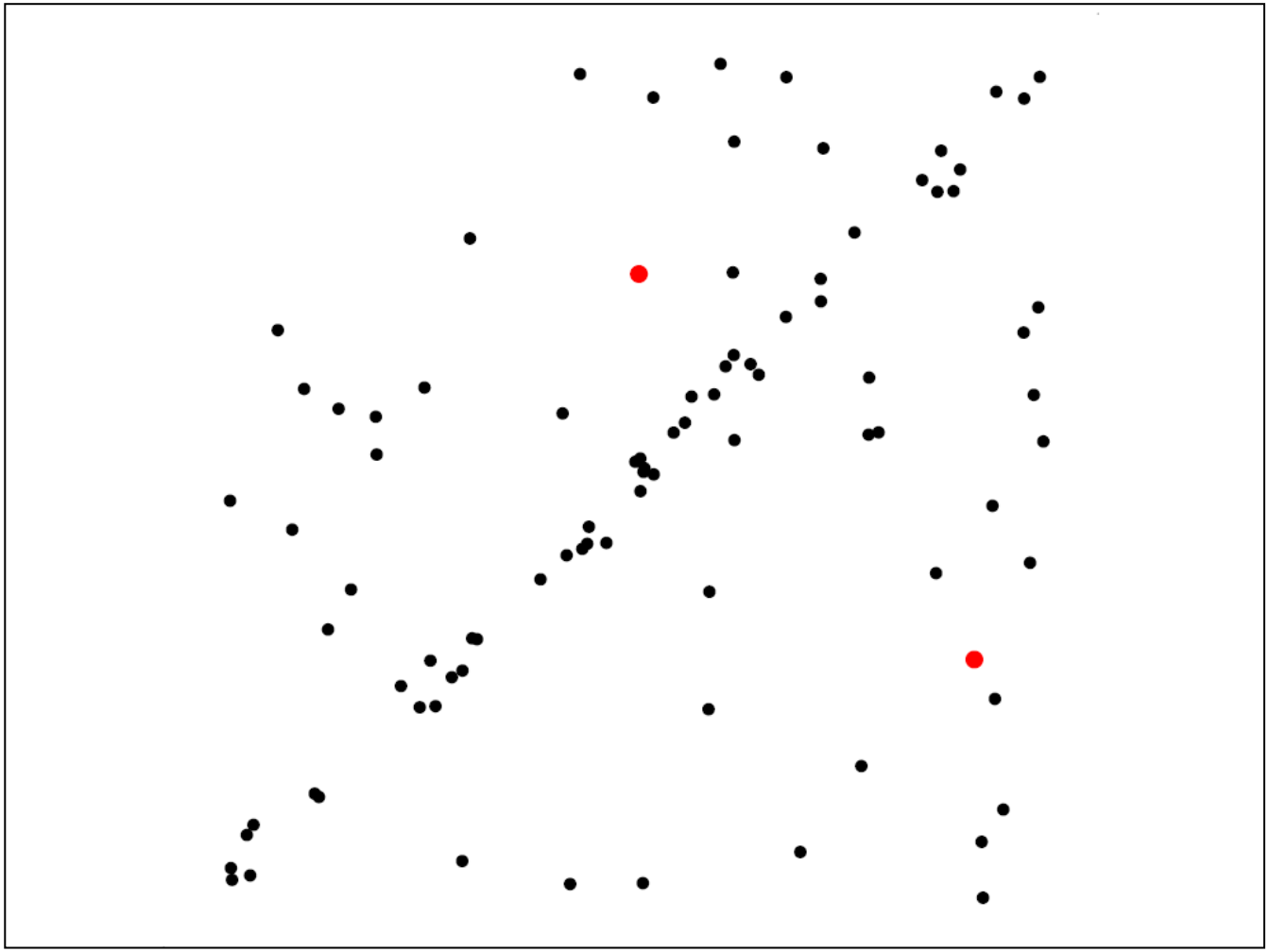
RANSAC pentru detectarea liniilor

Soluția metodei celor mai mici pătrate



Slide adaptat după S. Lazebnik

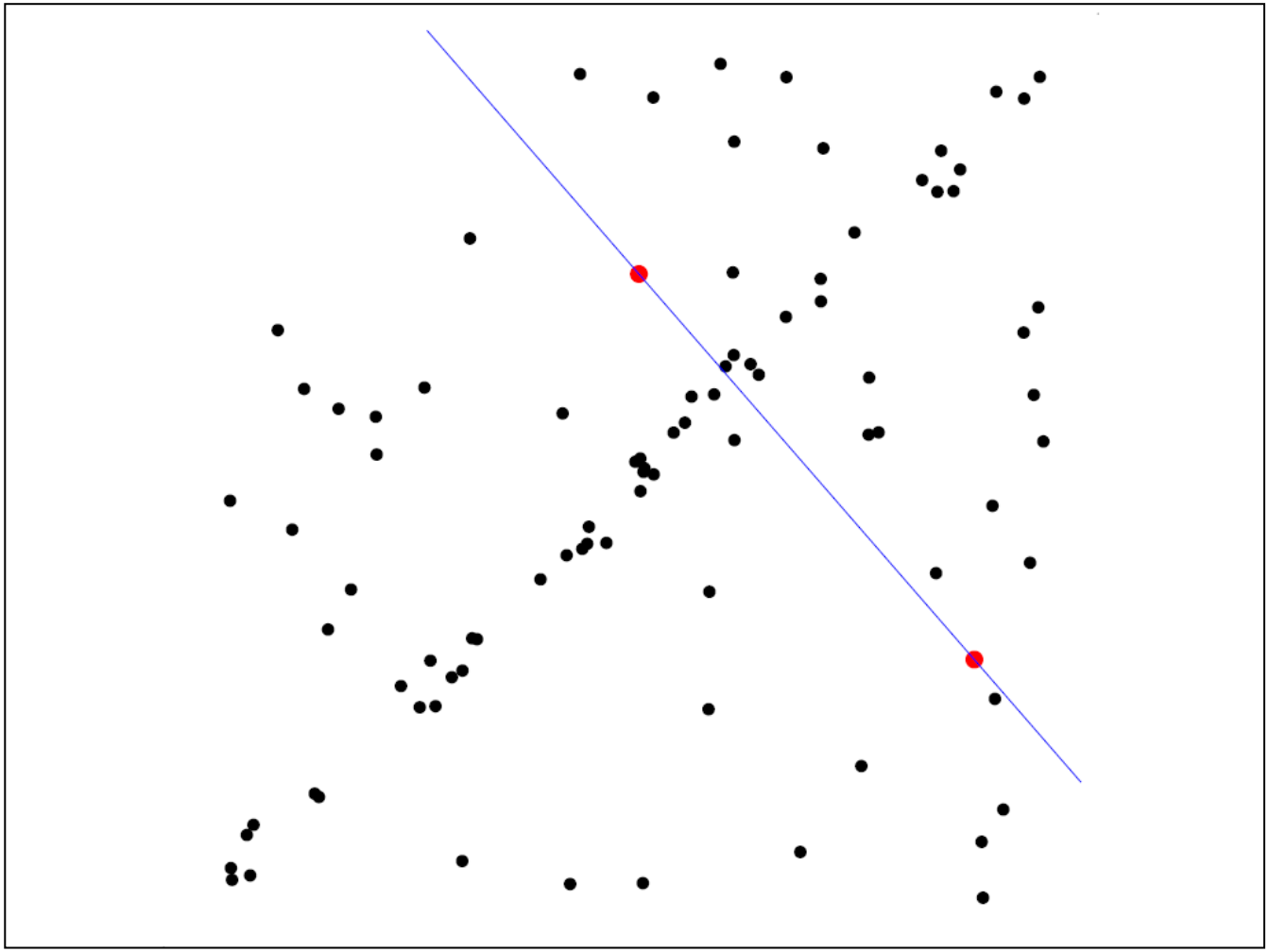
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate

Slide adaptat după S. Lazebnik

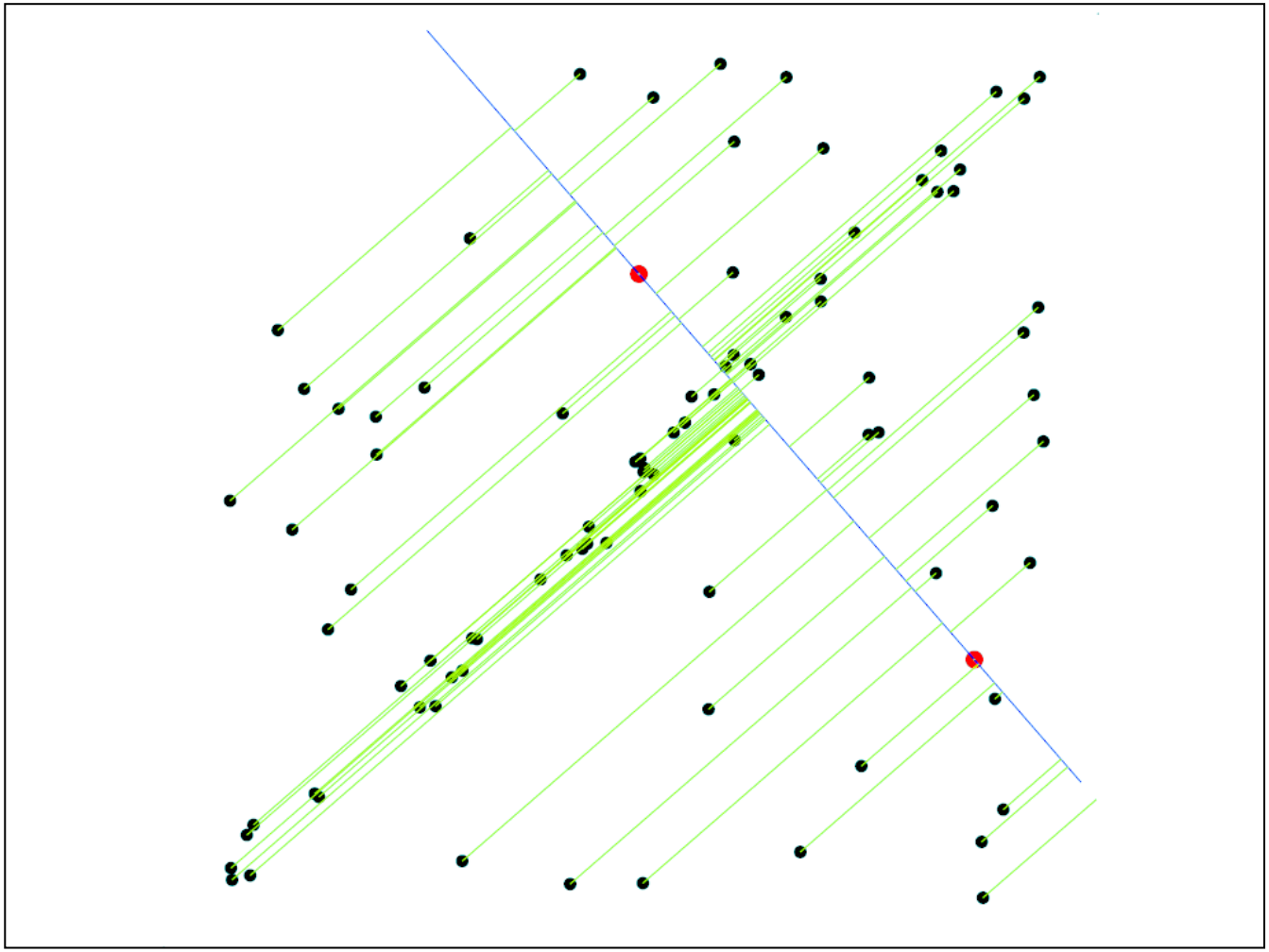
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta

Slide adaptat după S. Lazebnik

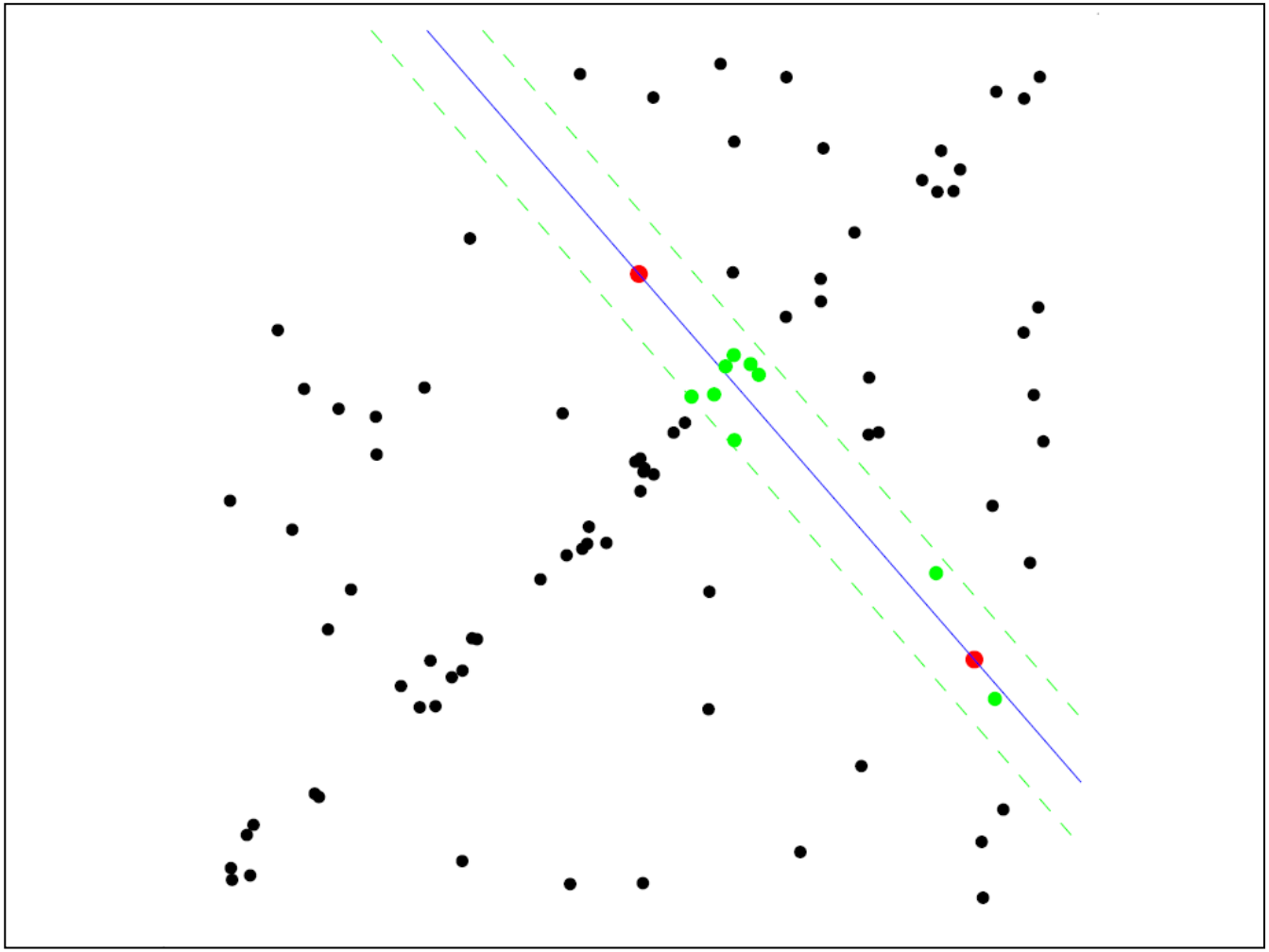
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare

Slide adaptat după S. Lazebnik

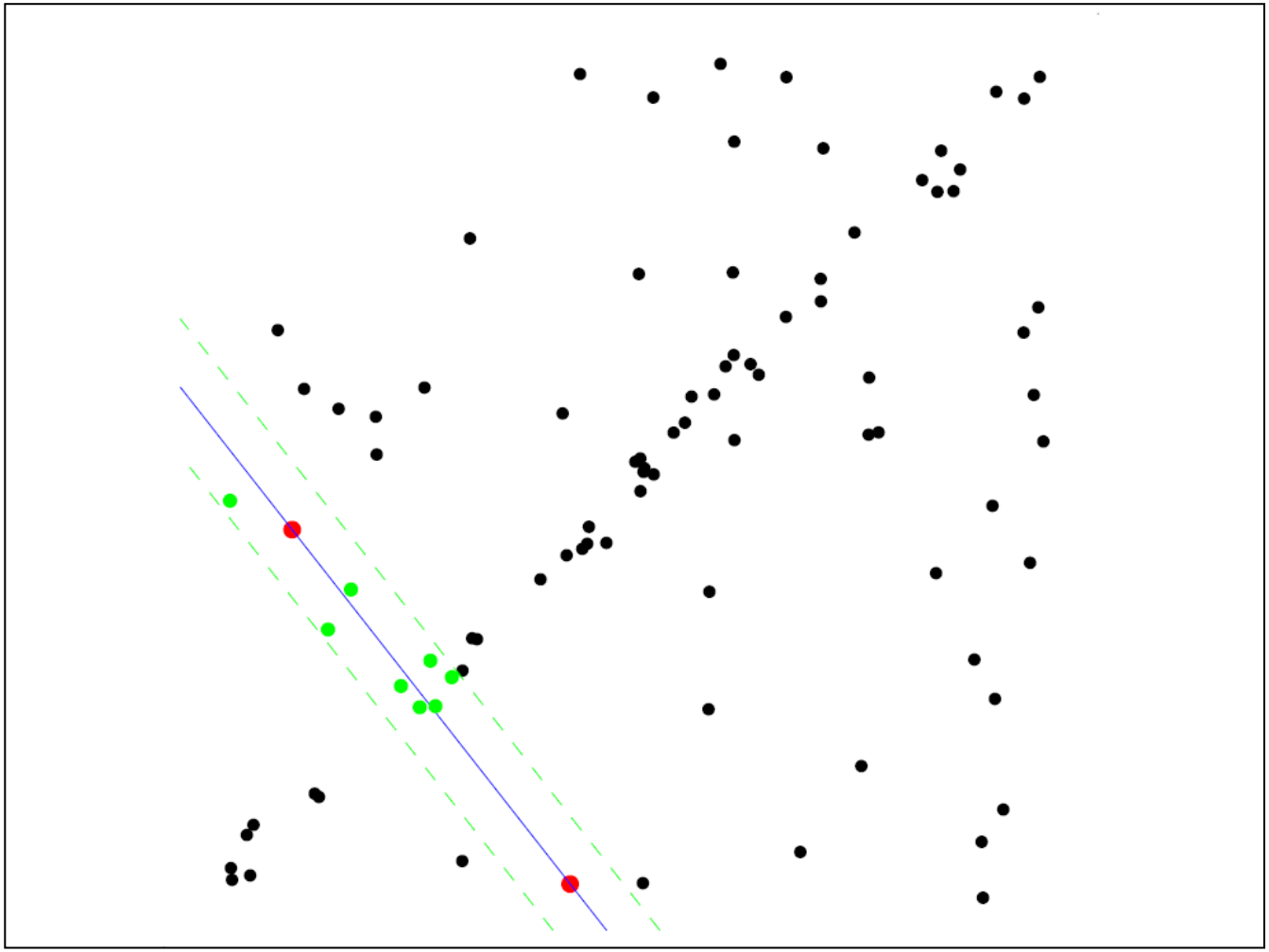
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistente

Slide adaptat după S. Lazebnik

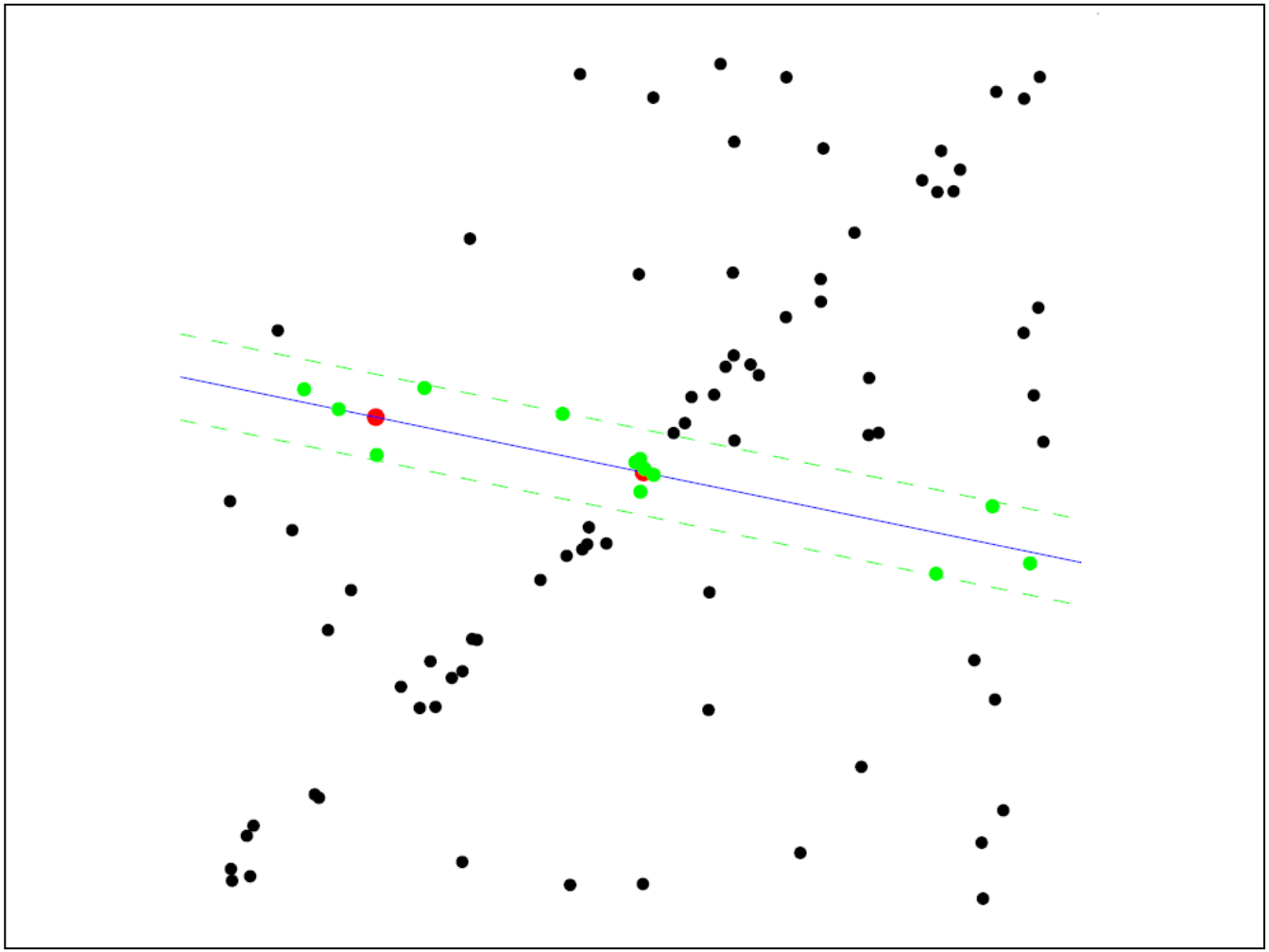
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistente
5. Repetă 1-4 pentru un număr de ori dat

Slide adaptat după S. Lazebnik

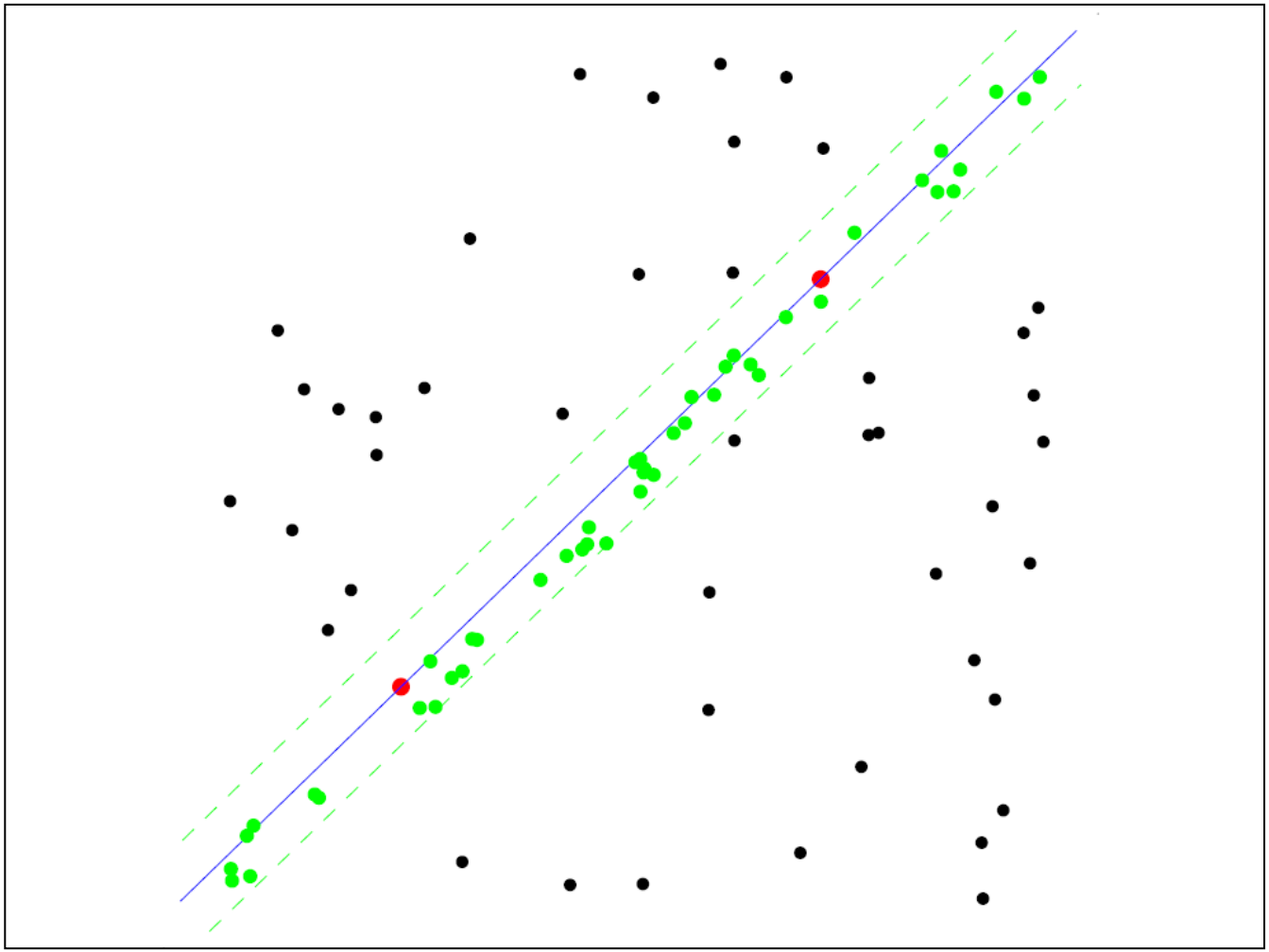
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistente
5. Repetă 1-4 pentru un număr de ori dat

Slide adaptat după S. Lazebnik

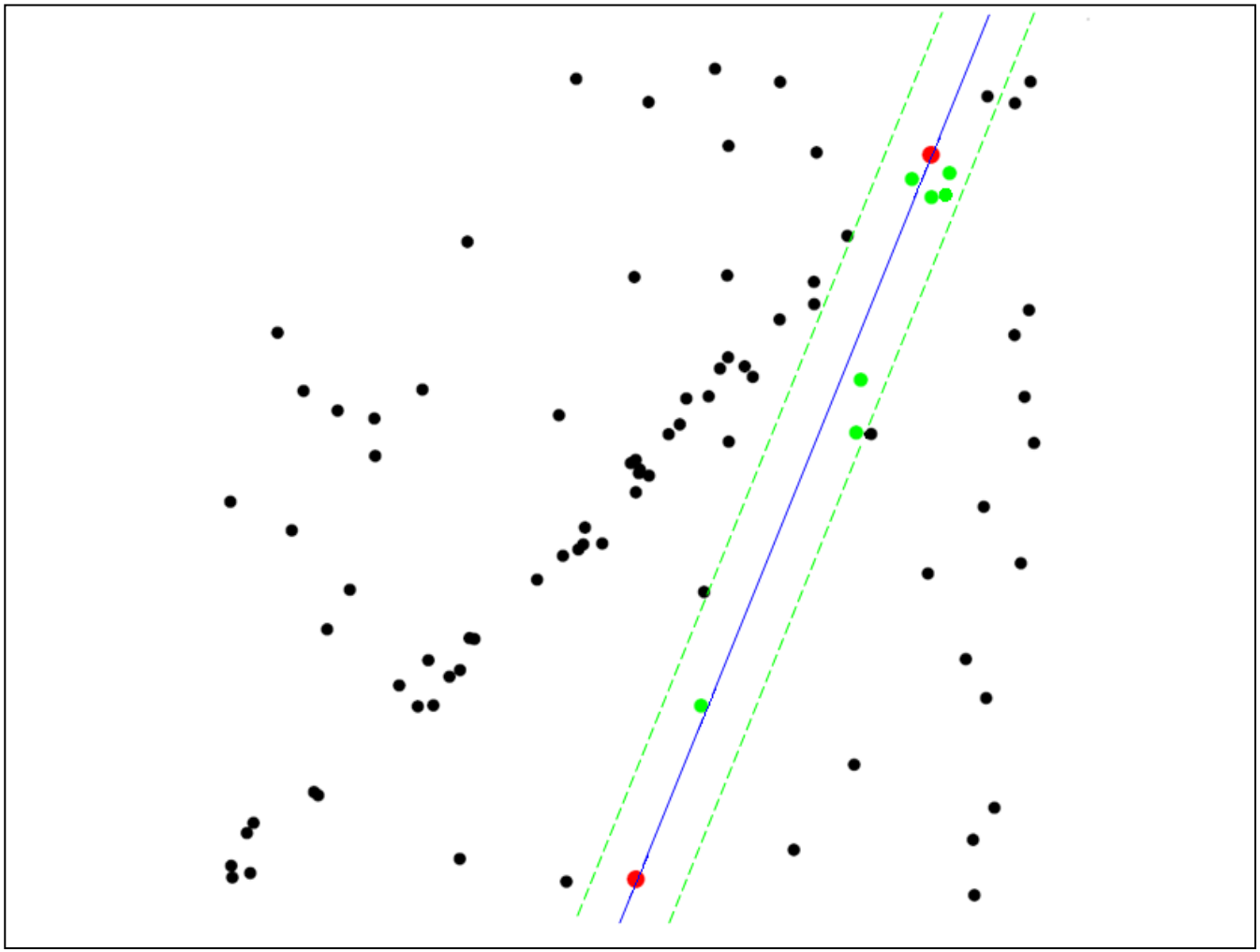
RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistente
5. Repetă 1-4 pentru un număr de ori dat

Slide adaptat după S. Lazebnik

RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistente
5. Repetă 1-4 pentru un număr de ori dat

Slide adaptat după S. Lazebnik

RANSAC pentru detectarea liniilor

- RANdom SAmple Consensus

- **idee principală**: vrem să evităm impactul punctelor outlier, ne concentrăm să găsim punctele “inlier”, și le folosim doar pe acestea la găsirea parametrilor modelului

- **intuiție**: dacă un punct outlier este inclus în dreaptă, numărul de puncte inlier va fi foarte mic (ipoteza nu este susținută de date)

RANSAC pentru detectarea liniilor

Repetă de ***N*** ori:

1. Selectează ***2*** puncte la întâmplare

2. Găsește ecuației dreptei care trece prin aceste două puncte

3. Găsește punctele inlier pentru această ipoteză din cele rămase: punctele a căror distanță față de dreaptă este < ***t*** (threshold)

4. Dacă există mai mult de ***d*** puncte inlier, acceptă ipoteza. Rafinează ipoteza pe baza tuturor punctelor inlier.

(Păstrează ipoteza cu cea mai mare număr de puncte inlier)

Algoritmul RANSAC: avantaje și dezavantaje

Avantaje

- simplu și general
- aplicabil în multe probleme (detectarea liniilor, construirea de panorame: găsirea de corespondențe – calculul homografiei - 8 parametri)
- funcționează destul de bine în practică

Dezavantaje

- numărul de parametri ai unui model poate fi foarte mare
- nu merge bine când numărul de puncte inlier este foarte mic (raportul inlier/outlier este mic)
- numărul minim de puncte ce definește un model nu furnizează întotdeaună ipoteze bune

Algoritmul RANSAC: ce N alegem?

- de câte ori **N** trebuie să repetăm RANSAC pentru a fi siguri (cu o probabilitate mare) că am găsit cea mai bună ipoteză?
- presupunem că procentul de puncte provenite din dreaptă din numărul total de puncte este **r (= #puncte inlier/nr puncte total)**
- o ipoteză este definită de **k** puncte (**k=2** pentru dreaptă)
- probabilitea ca o selecție aleatoare de n puncte să fie corectă $\mathbf{r^k}$
- probabilitatea ca toate selecțiile să fie greșite: $\mathbf{(1 - r^k)^N}$
- alege **N** astfel încât $p = 1 - \mathbf{(1 - r^k)^N}$ este foarte mare (1-0.01=0.99)

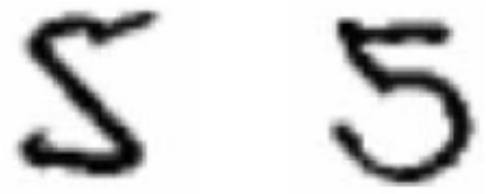
RANSAC: Calculul lui N (p=0.99)

k	Proporția de outlieri (1-r)						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Slide adaptat după D. Lowe

Compararea contururilor

Compararea contururilor



Exemple de două imagini cu cifre scrise de mână.
Comparând pixel cu pixel, cele două imagini sunt foarte diferite.
Totuşi, în termeni de formă, cele două imagini sunt similare.

Slide adaptat după S. Belongie

Distanța Chamfer

- Distanța medie dintre punctele unui ***template T*** şi o mulțime de puncte (e.g. ***puncte cu intensitate = 0***, puncte cu anumite caracteristici) dintr-o imagine I.

$$d_{chamfer}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

template

$I =$ mulțime de puncte dintr-o imagine

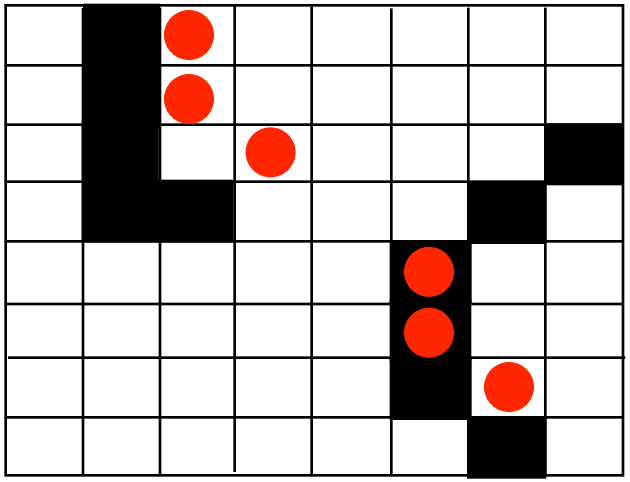
$T =$ mulțime de puncte ale unui ***template***

$d_I(t) =$ distanța minimă dintre ***punctul t*** şi un punct din I (Manhattan, Euclidiană)

Slide adaptat după K. Grauman

Distanța Chamfer

Puncte din I (2D)

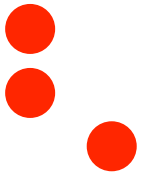


Distanța Manhattan:

$$\frac{1}{3}(1+1+2) = 1.33$$

$$\frac{1}{3}(0+0+1) = 0.33$$

T



$$d_{chamfer}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

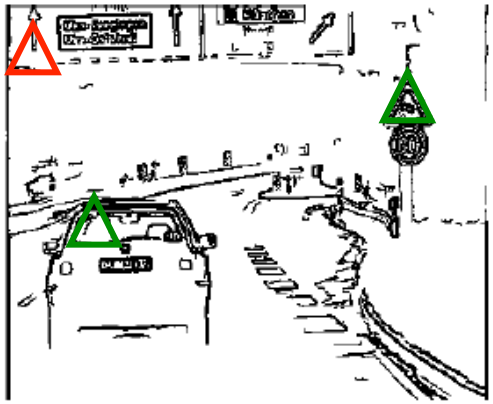
Slide adaptat după K. Grauman

Distanța Chamfer

- distanța medie până la cel mai apropiat punct din ***I***

$$d_{chamfer}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

Cum diferă distanța Chamfer de filtrarea cu o mască de forma lui T?



Imagine cu muchii

Răspunsul variază mult mai puțin abrupt decât în cazul filtrării! (dacă mă mut la stânga/dreapta cu un pixel am valori apropiate)

Implementare naivă – complexitate?

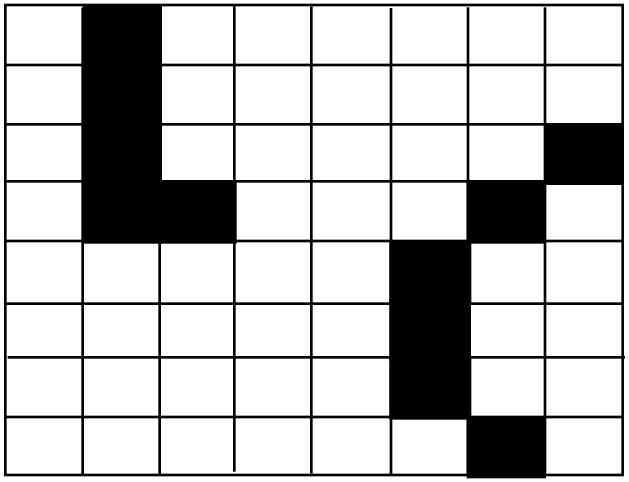
$|T| * |I|^2$ operații

Maxime locale ale funcției care calculează distanța Chamfer pentru fiecare fereastră

Slide adaptat după K. Grauman

Transformata Distanță (distance transform)

Puncte din I (2D)



Transformata Distanță

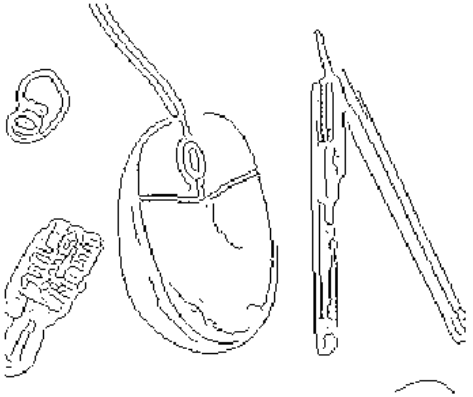
1	0	1	2	3	4	3	2
1	0	1	2	3	3	2	1
1	0	1	2	3	2	1	0
1	0	0	1	2	1	0	1
2	1	1	2	1	0	1	2
3	2	2	2	1	0	1	2
4	3	3	2	1	0	1	2
5	4	4	3	2	1	0	1

Transformata Distanță (TD) este o funcție care pentru fiecare pixel *p* asignează un număr pozitiv *TD(p)* corespunzând distanței de la *p* la cel mai apropiat punct din mulțimea *I*

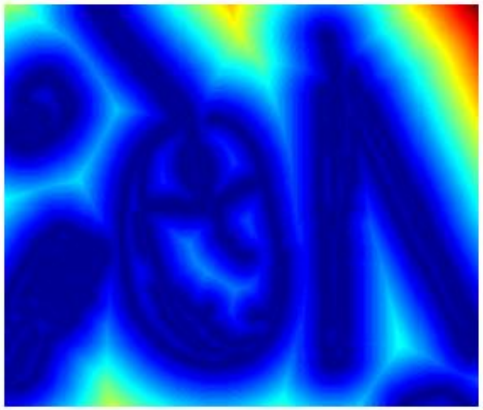
Slide adaptat după Y. Boykov

Transformata Distanță

imagine inițială



muchii



transformata distanță

Calculăm o singură dată TD(), apoi citim din acest tabel distanța la cel mai apropiat punct de pe muchie.

Valoarea la (x,y) indică cât de departe (x,y) este de cel mai apropiat punct de pe muchie
albastru – valori mici,
roșu – valori mari

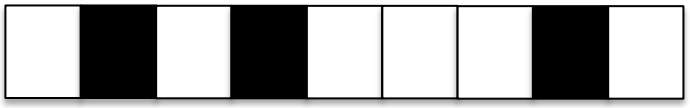
MATLAB: bwdist

Slide adaptat după K. Grauman

Transformata Distanță – 1D

Două treceri de complexitate O(n)

Exemplu



1. Inițializare

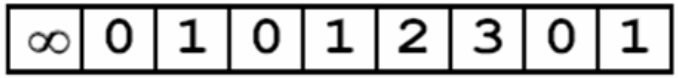
$$TD(j) = \begin{cases} 0, & \text{dacă j este în I} \\ \infty, & \text{altfel} \end{cases}$$

for j = 1:n

end

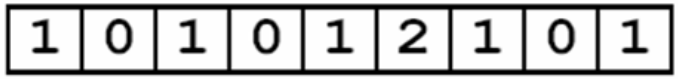
2. Trecere înainte (vecin stânga)

```
for j = 2:n
        TD(j) = min(TD(j),TD(j-1)+1)
end
```



3. Trecere înapoi (vecin dreapta)

```
for j = n-1:-1:1
        TD(j) = min(TD(j),TD(j+1)+1)
end
```

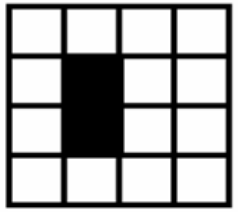


Slide adaptat după D. Huttenlocher

Transformata Distanță – 2D

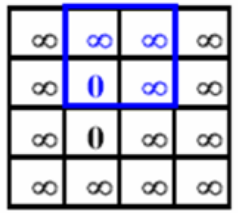
Analog cu 1D

Exemplu



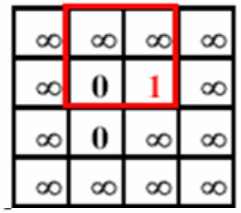
1. Inițializare

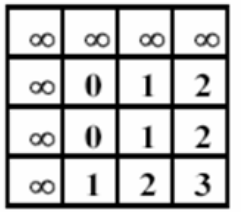
$$TD(i,j) = \begin{cases} 0, & \text{dacă } (i,j) \text{ este în I} \\ \infty, & \text{altfel} \end{cases}$$



2. Trecere înainte (vecin stânga, sus)

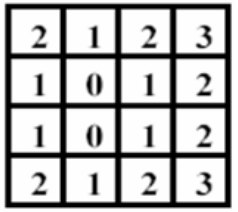
$$TD(i,j) = \min(TD(i,j), TD(i,j-1)+1, TD(i-1,j)+1)$$





3. Trecere înapoi (vecin dreapta, jos)

$$TD(i,j) = \min(TD(i,j), TD(i,j+1)+1, TD(i+1,j)+1)$$



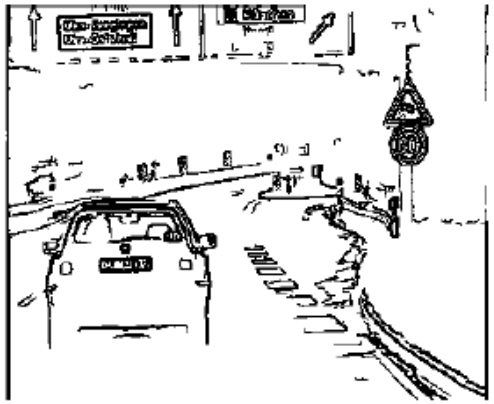
Slide adaptat după D. Huttenlocher

Distanța Chamfer

- distanța medie până la cel mai apropiat punct din ***I***

$$D_{chamfer} = (T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

Complexitate implementare naivă:
$|T| * |I|^2$ operații
Complexitate transformata distanță:
$|T| * |I|$ operații



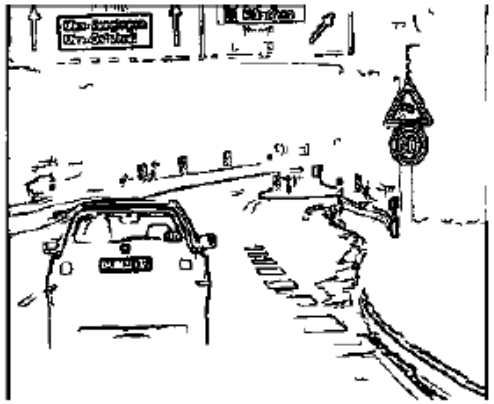
Imagine cu muchii

transformata distanță
alb – distanță mare,
negru - distanță mică

Slide adaptat după K. Grauman

Recunoaşterea semnelor de circulație

Recunoaștere = detectare + clasificare
Detectare = localizăm conturul în imagine (pe baza de template: triunghi, cerc)
Clasificare = analizăm pixelii din interiorul conturului și clasificăm semnul de circulație într-o clasă (limitare de viteză 60 km, limitare de viteză 80 km, semafor, etc)



Imagine cu muchii

transformata distanță alb – distanță mare, negru - distanță mică

Slide adaptat după K. Grauman

Recunoaşterea semnelor de circulație

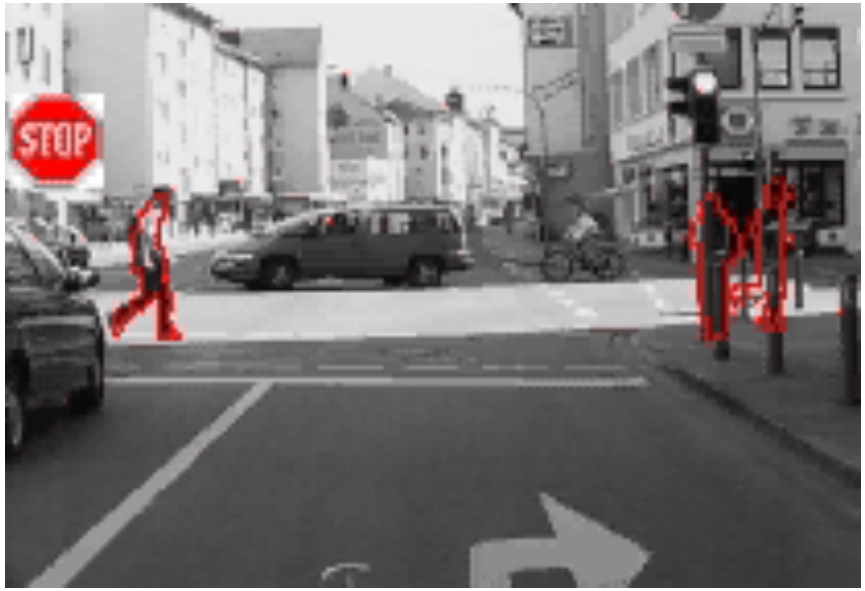
condiții de zi

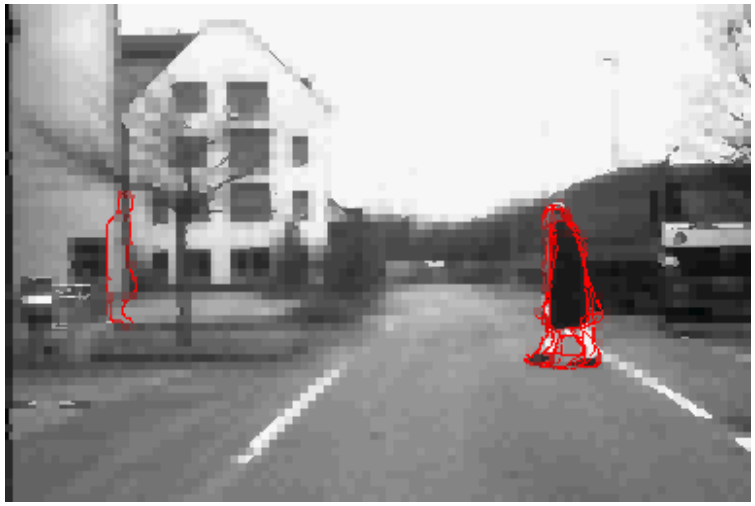
condiții de noapte

Video de la D. Gavrila, DAGM 1999

Detectarea pietonilor

Detectare pietonilor = localizăm contururi în imagine asemănătoare cu contururi de pietoni dintr-o mulțime de template-uri





http://www.gavrila.net/Research/Chamfer_System/chamfer_system.html

Distanța Chamfer: proprietăți

- Avantaje
  - simplă de implementat
  - foarte rapidă
  - tolerantă la forme care diferă puțin de template

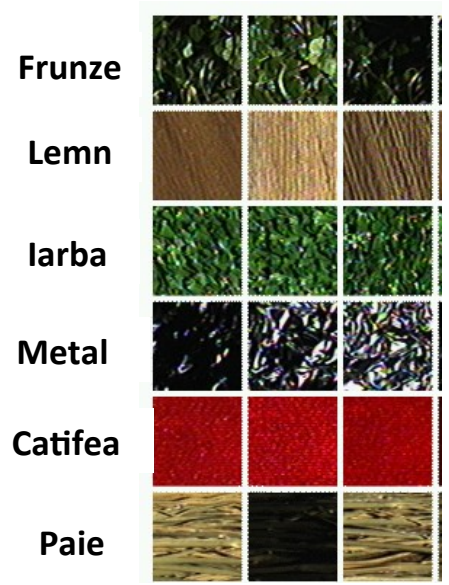
- Dezavantaje
  - senzitivă la mărime (scale) şi rotație
  - număr mare de templaturi pentru forme diverse ca mărime şi ca rotație
  - detecții false în regiuni cu multe muchii

Slide adaptat după K. Grauman

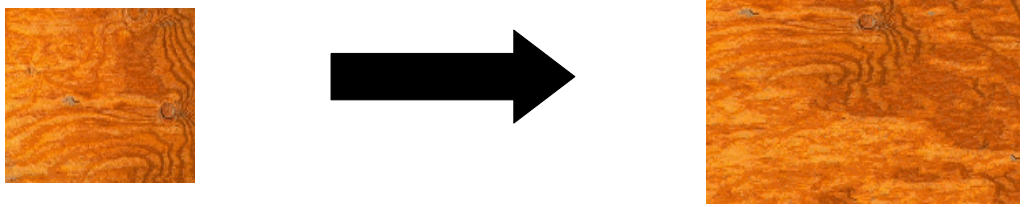
Textură

Textură

- Analiza texturii:
  - reprezentăm şi comparăm texturile pentru a decide dacă sunt la fel



- Sinteza texturii :
  - fiind dată o mostră de textură (dimensiuni mici) vrem să generăm o textură similară (dimensiuni mari)



mostră de textură

textură generată

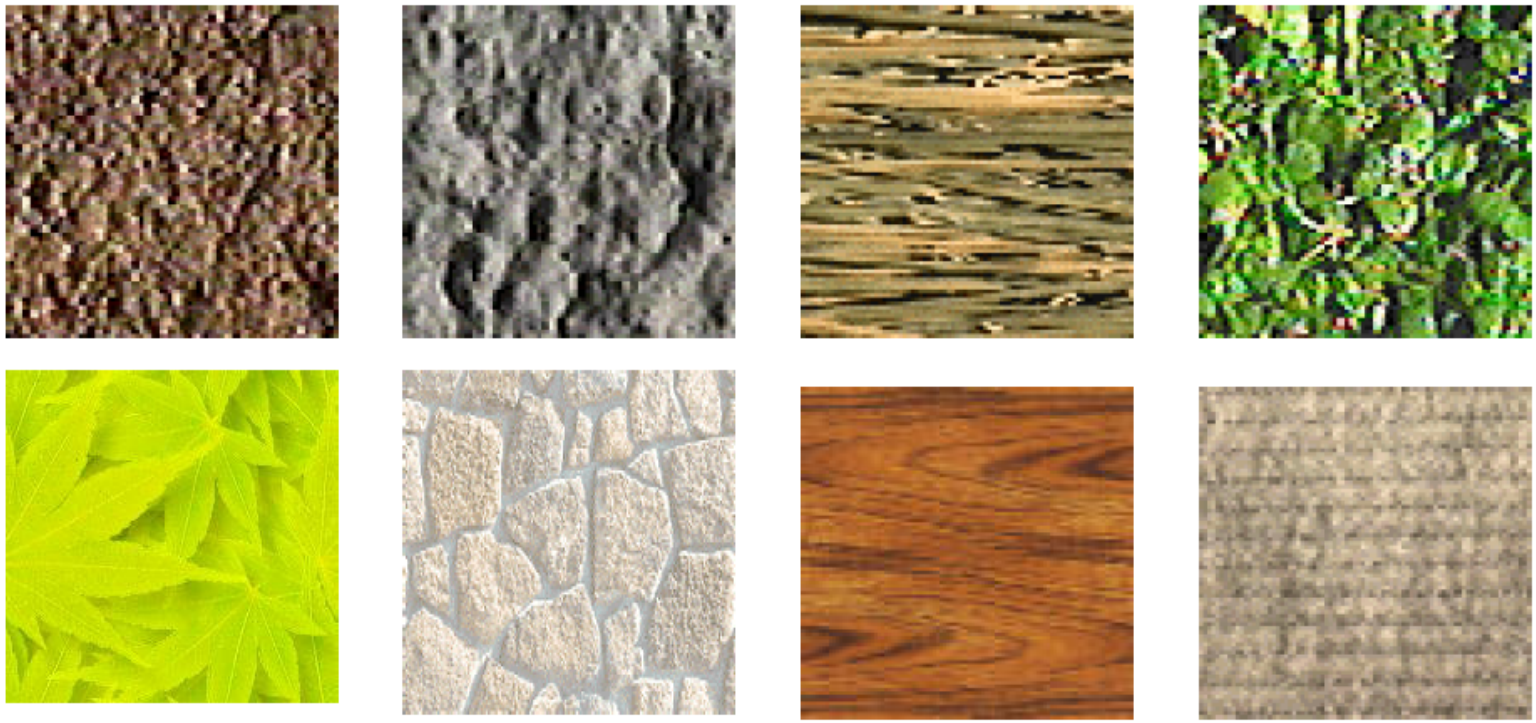
Idei pt licență: segmentarea imaginilor aeriene prin analiza texturii

Sinteza texturii (manuală)

Sinteza texturii (manuală)

http://www.dailykos.com/story/2004/10/27/22442/878

Textură - exemple

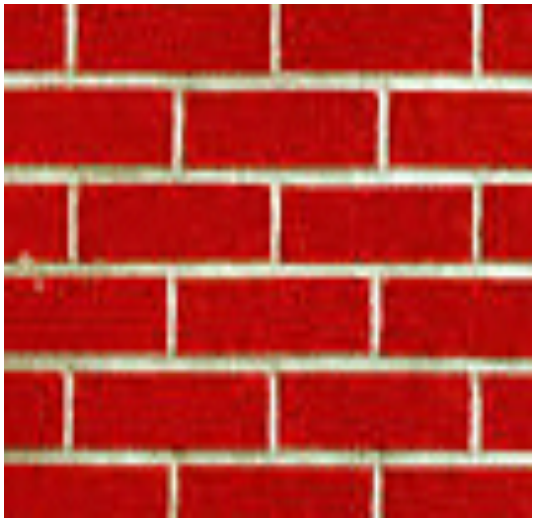
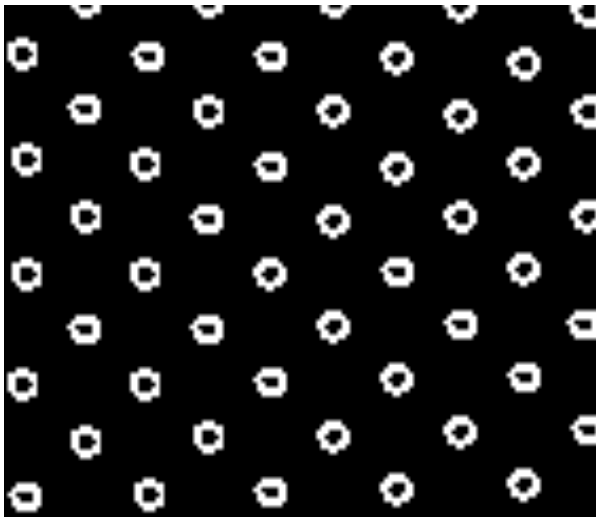
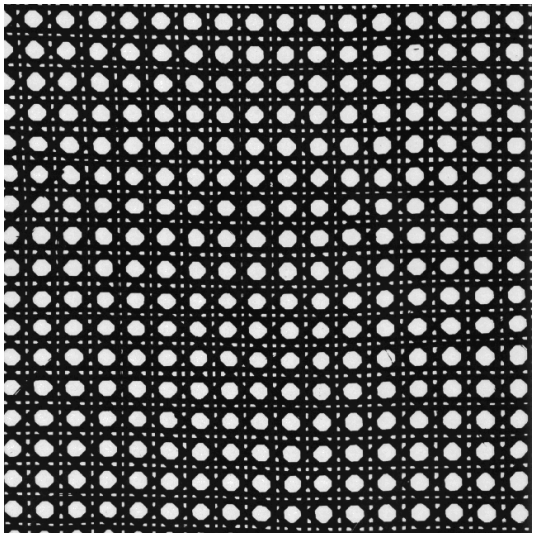


Ce defineşte o textură?

Structuri similare de pixeli (pattern-uri) care se repetă

Satisfac anumite proprietăți statistice

Textură - exemple

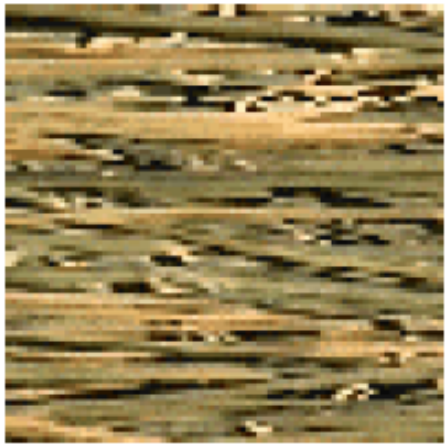
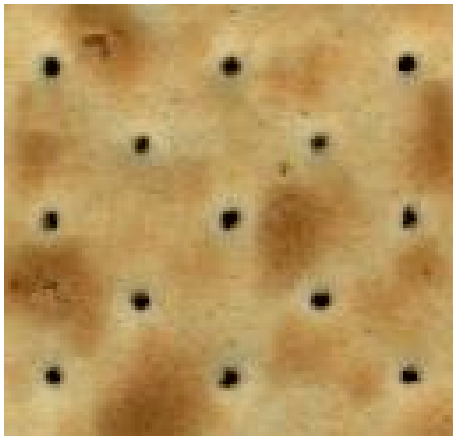




Ce diferă față de exemplul anterior?

Pattern-urile acestor texturi sunt mult mai regulate

Textură - exemple



Ce diferă față de exemplul anterior?
Pattern-urile acestor texturi sunt mai puțin regulate (aleatoare, stocastice)

Mărime: obiect vs. textură

Acelaşi lucru în lumea reală poate apărea ca obiect sau ca o textură, în funcție de mărimea pe care o considerăm.

De ce studiem textura?

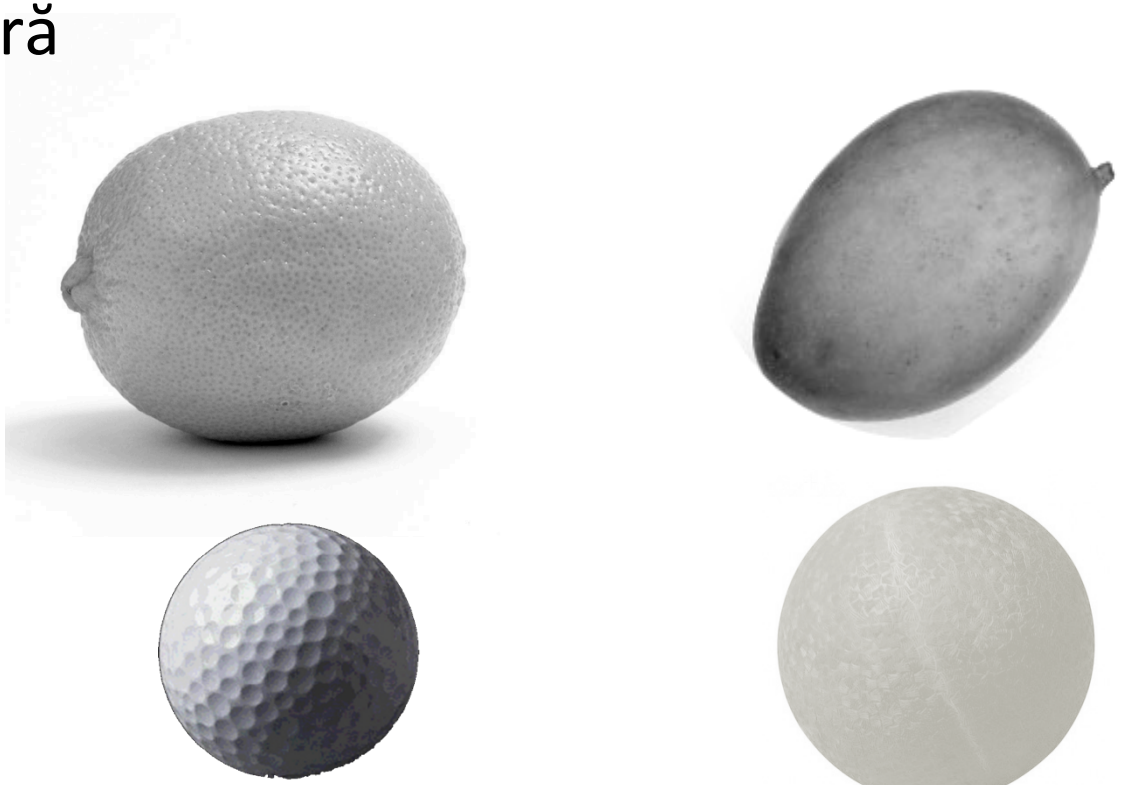
Importantă pentru percepție:

- proprietățile materialelor

De ce studiem textura?

Importantă pentru percepție:

- proprietățile materialelor
- caracteristică discriminativă pentru obiecte cu formă similară



Slide adaptat după K. Grauman

De ce studiem textura?

Importantă pentru percepție:

- proprietățile materialelor
- caracteristică discriminativă pentru obiecte cu formă similară

Slide adaptat după K. Grauman

De ce studiem textura?

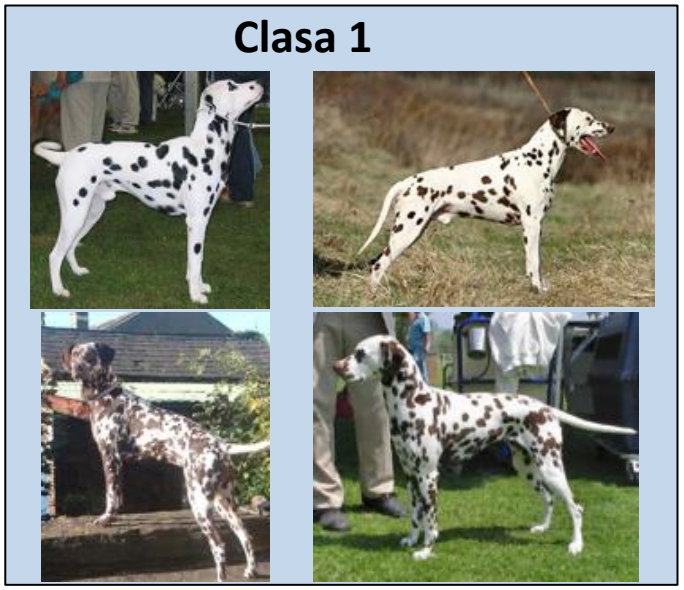
Importantă pentru percepție:

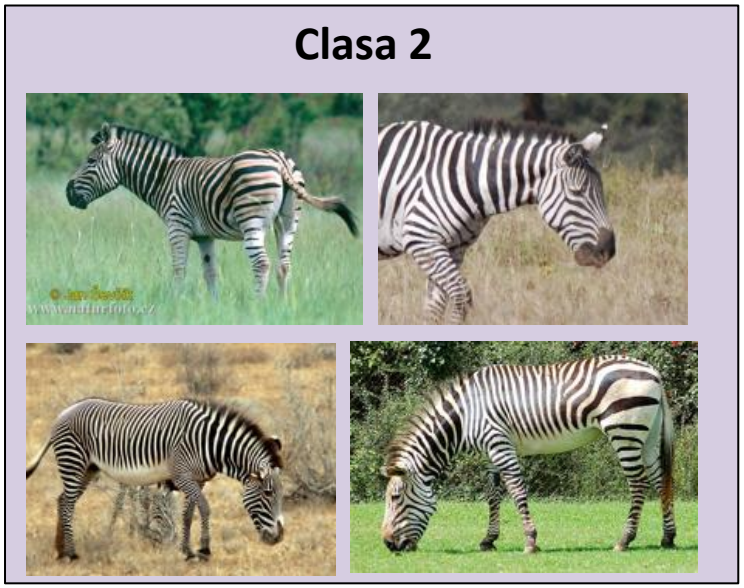
- proprietățile materialelor
- caracteristică discriminativă pentru obiecte cu formă similară
- caracteristică discriminativă pentru obiecte cu aceeași culoare

De ce studiem textura?

- **Segmentare**
  - grupăm (segmentăm) regiunile din imagine cu aceeaşi textură

- **Clasificare**
  - clasificăm obiecte ca aparținând unei clase de obiecte





Clasa ?

De ce studiem textura?

- **Segmentare**
  - grupăm (segmentăm) regiunile din imagine cu aceeaşi textură
- **Clasificare**
  - clasificăm obiecte ca aparținând unei clase de obiecte
- **Sinteză**
  - dată o mostră de textură (dimensiuni mici) vrem să generăm o textură similară (dimensiuni mari)

mostră de textură

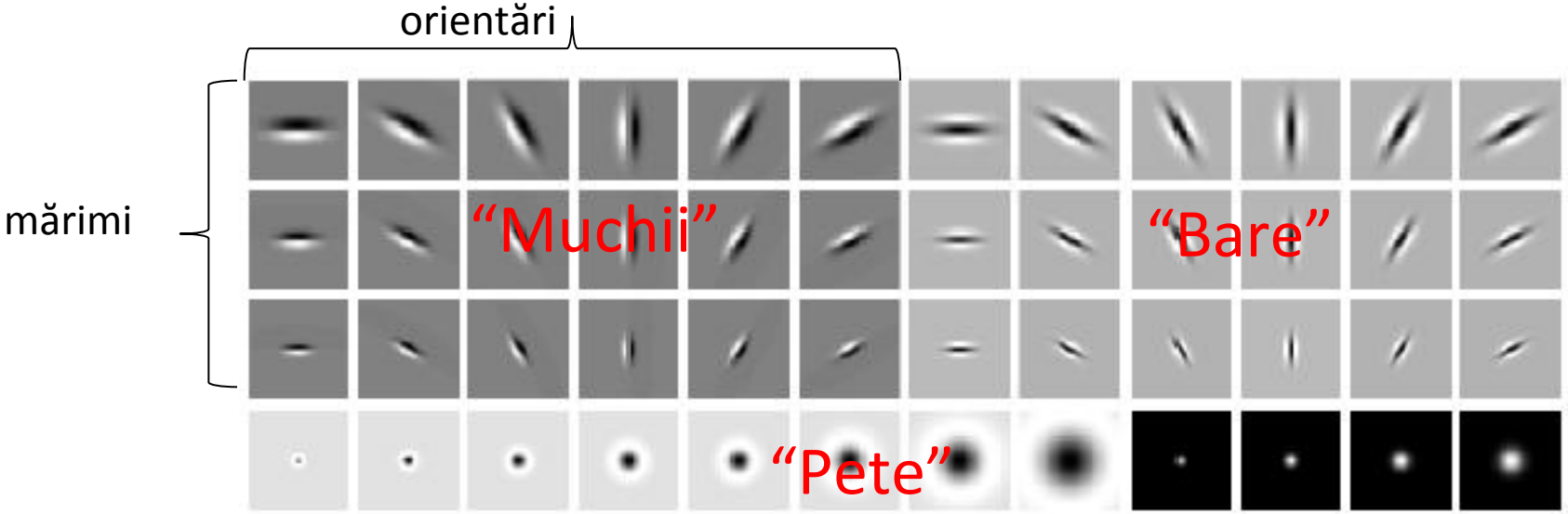
textură generată

Reprezentarea texturii

Texturile sunt formate din pattern-uri locale ce se repetă, aşa încât:

- găseşte aceste pattern-uri (texteli)
  - foloseşte filtre care arată ca pattern-urile
  - reține valoarea răspunsului după filtrare
- descrie datele d.p.d.v. statistic pentru fiecare fereastră locală
  - medie, deviație standard
  - histograme

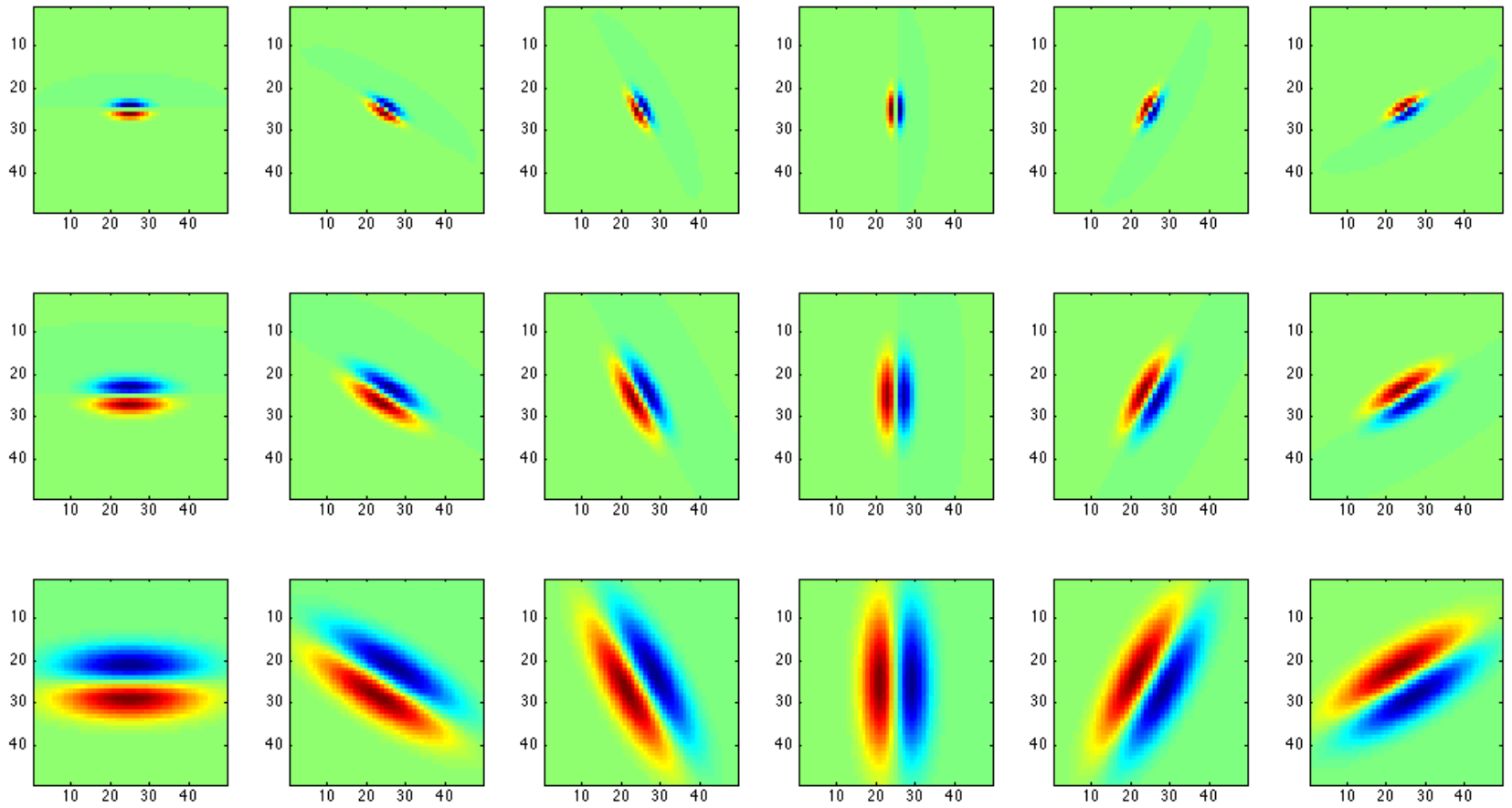
Colecții de filtre



- Ce filtre să conțină colecția?
  - o combinație de mărimi si orientări, diverse tipuri de pattern-uri.

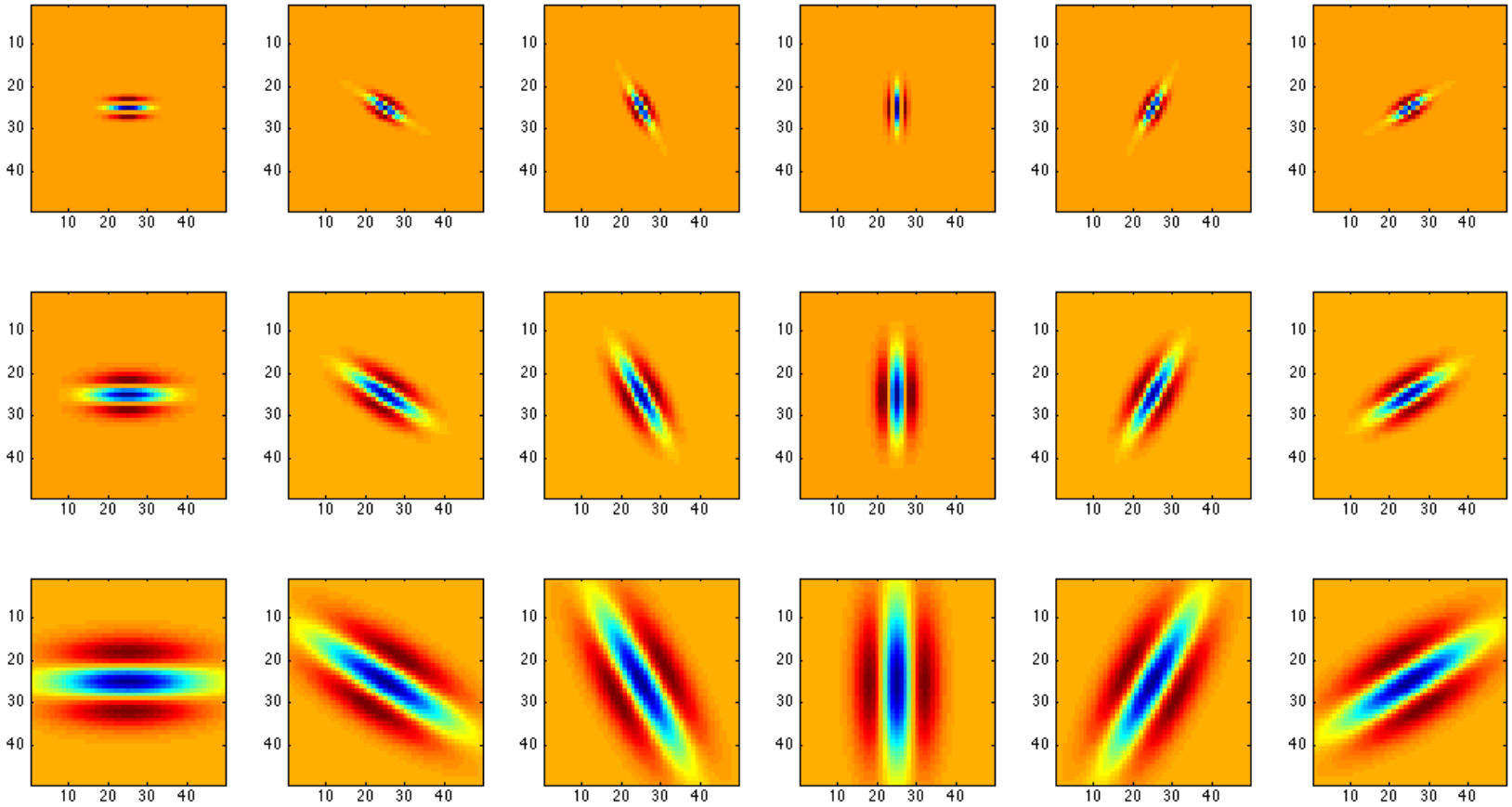
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Filtre pentru detectarea muchiilor



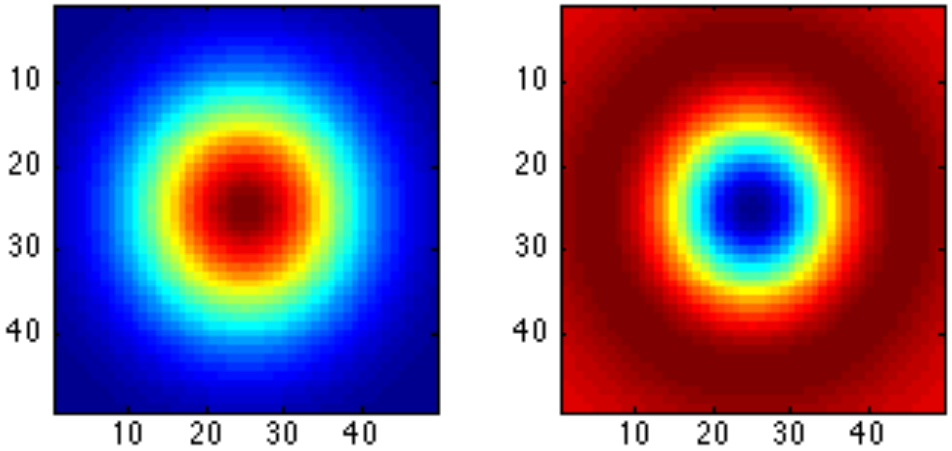
roșu – valori mari, albastru – valori mici

Filtru pentru detectarea barelor



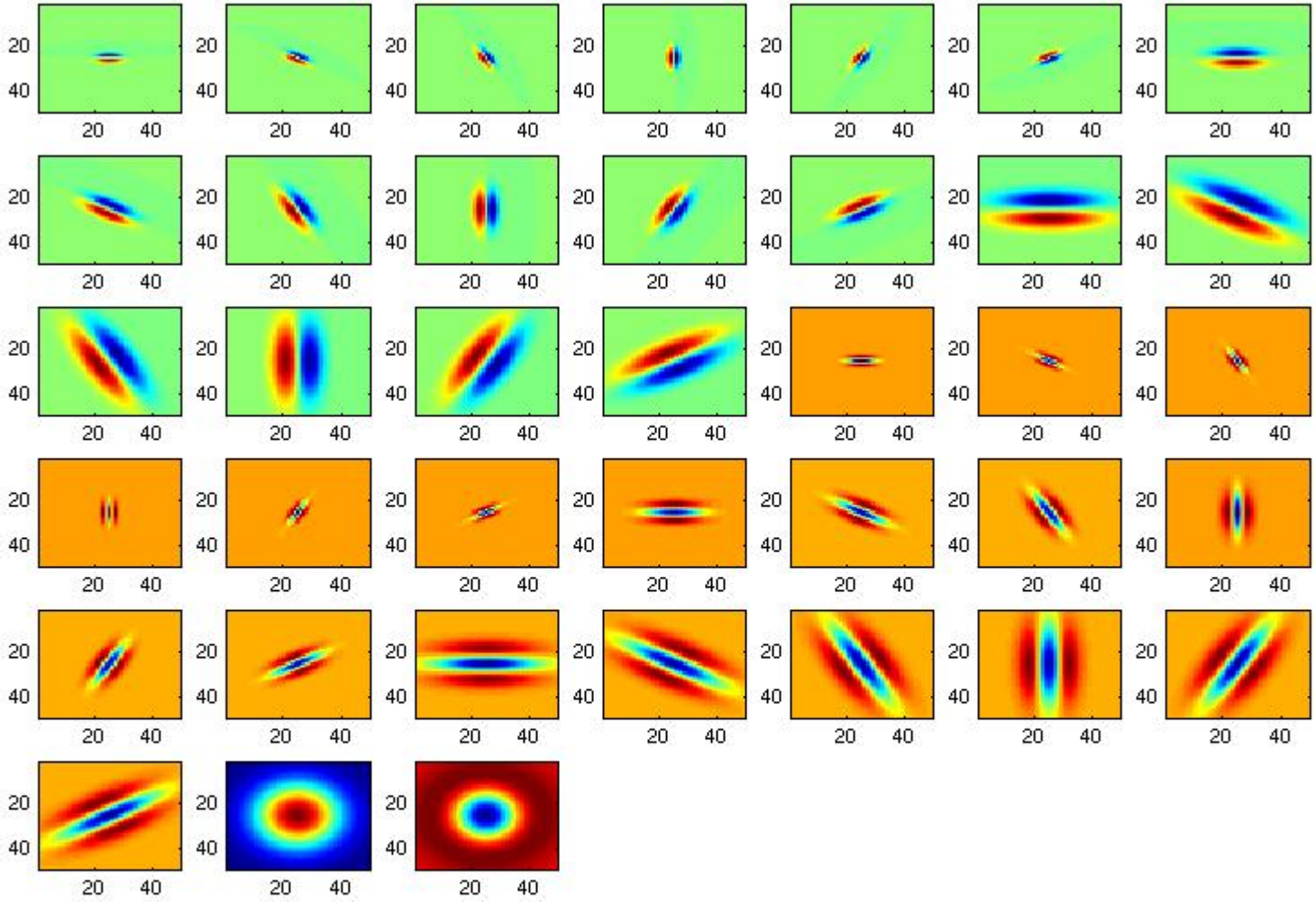
roșu – valori mari, albastru – valori mici

Filtru pentru detectarea petelor



roșu – valori mari, albastru – valori mici

Exemplu – colecția de filtre

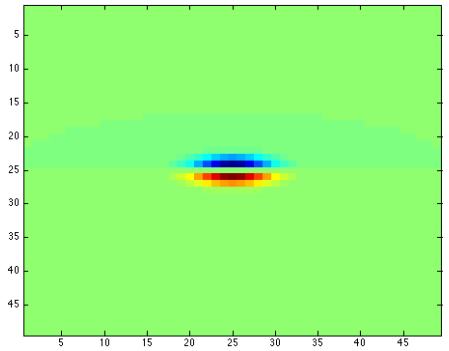


roșu – valori mari, albastru – valori mici

Imagini

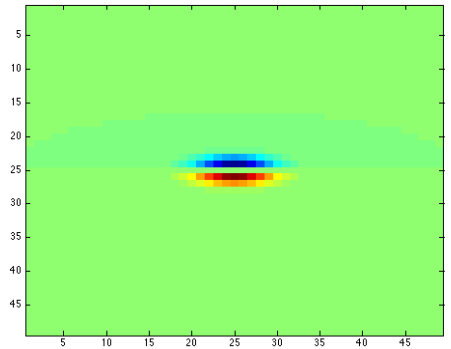
Slide adaptat după K. Grauman

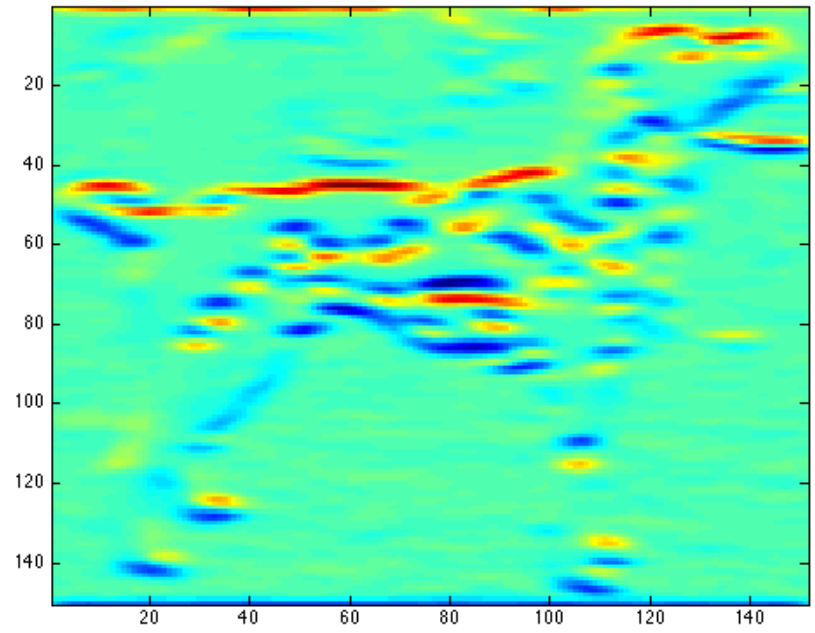
Imagini + filtru

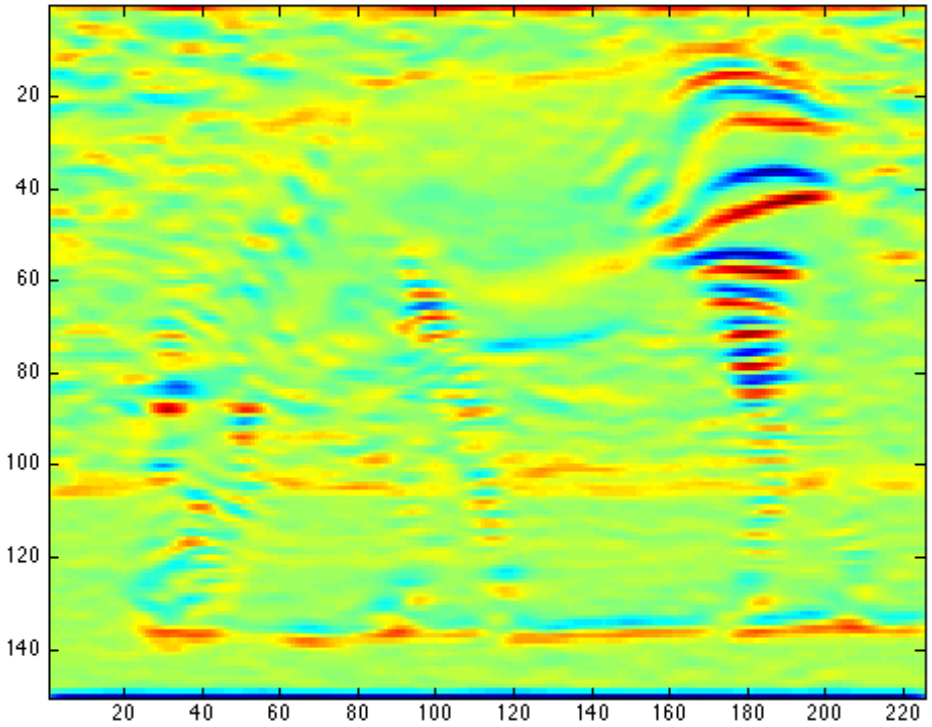


Slide adaptat după K. Grauman

Imagini filtrate

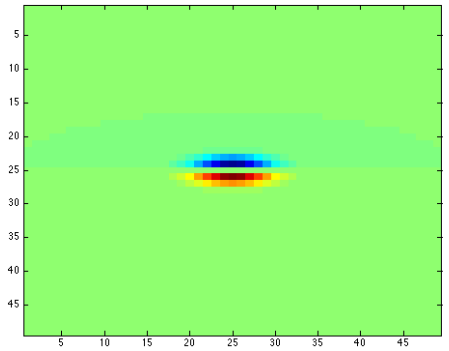


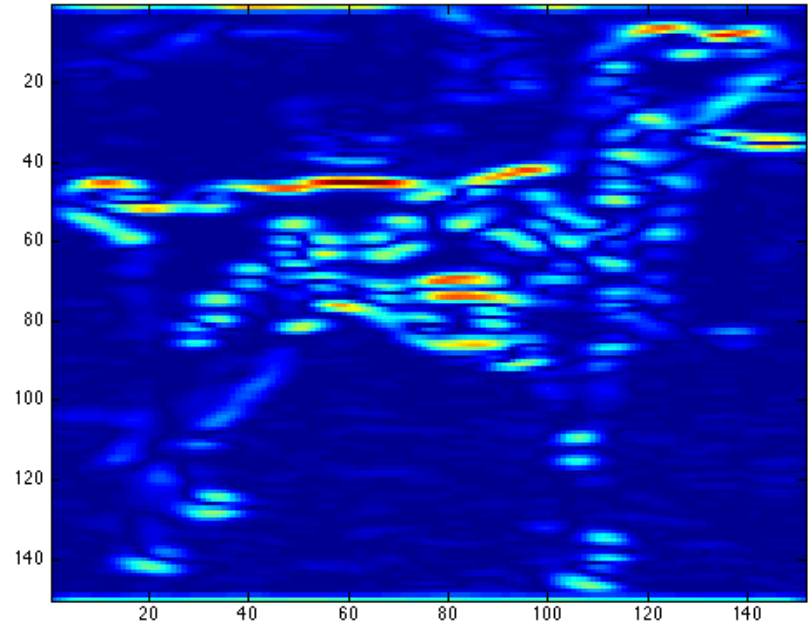


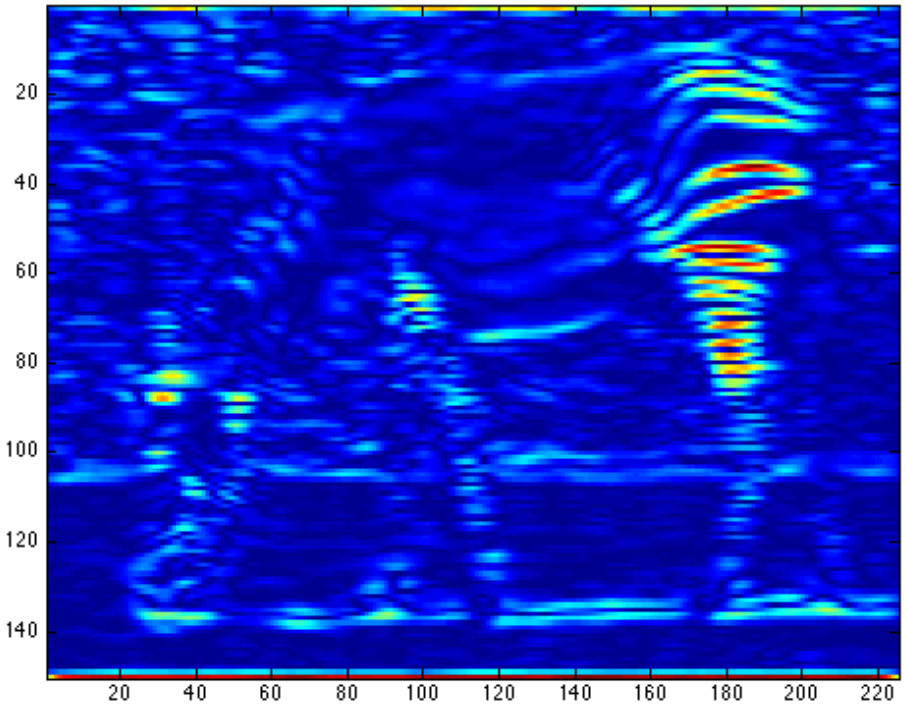


Slide adaptat după K. Grauman

Imagini filtrate
cu răspuns absolut

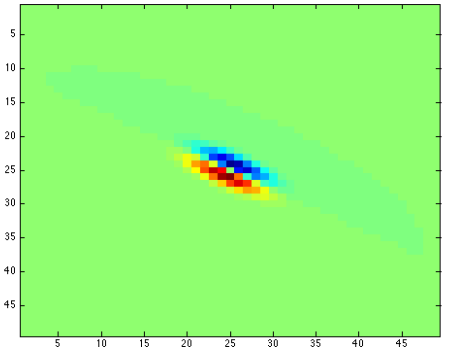






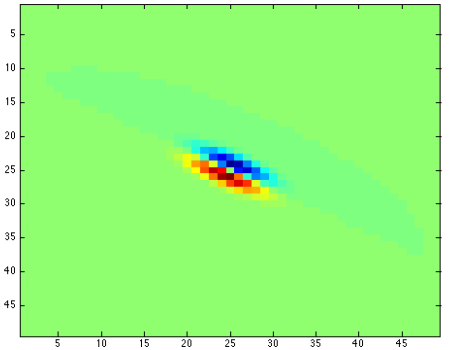
Slide adaptat după K. Grauman

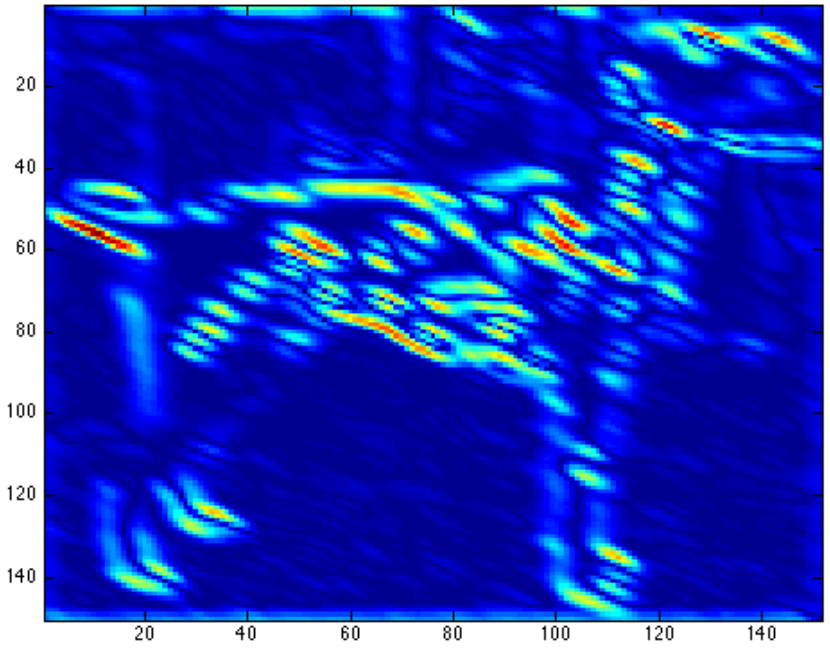
Imagini + filtru

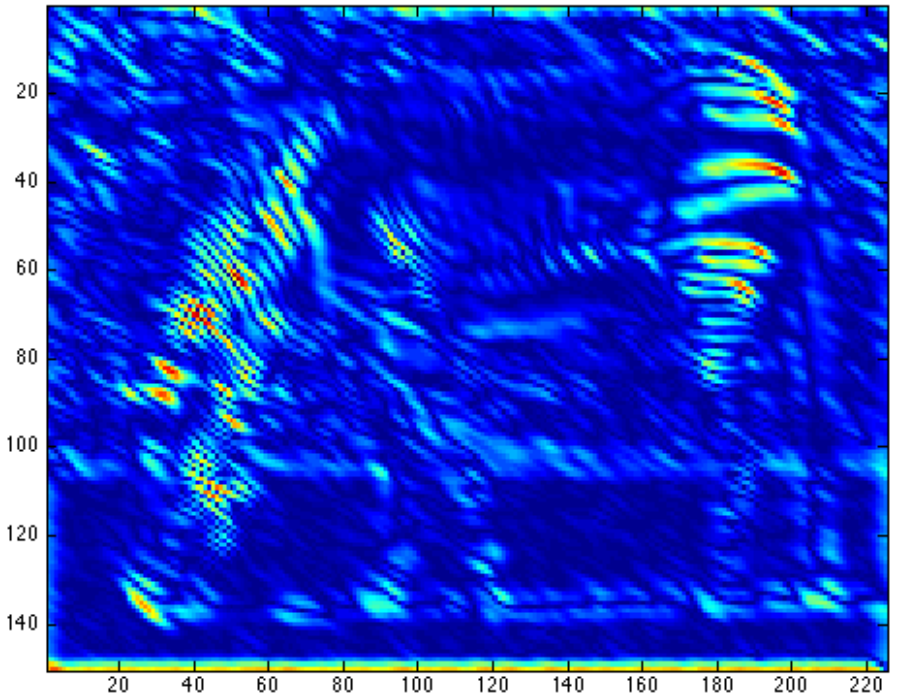


Slide adaptat după K. Grauman

Imagini filtrate cu răspuns absolut

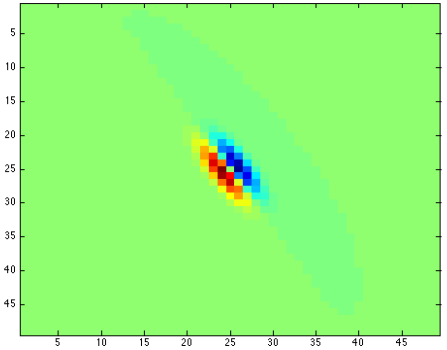






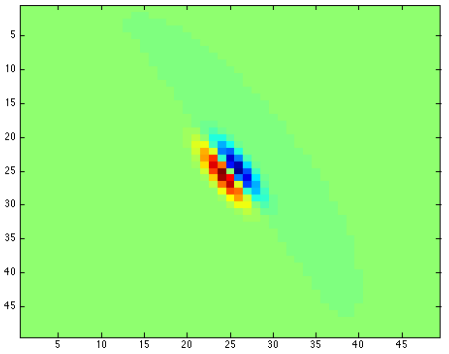
Slide adaptat după K. Grauman

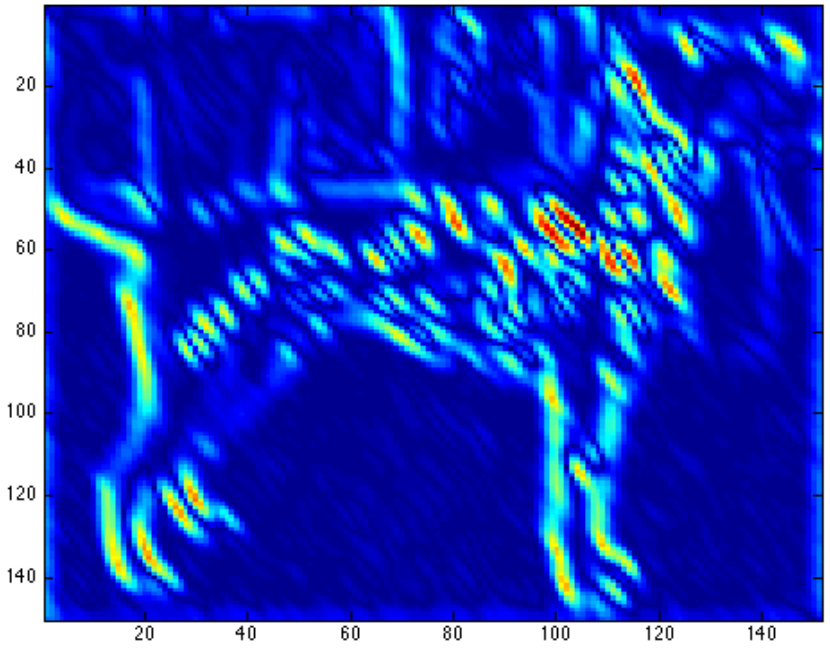
Imagini + filtru

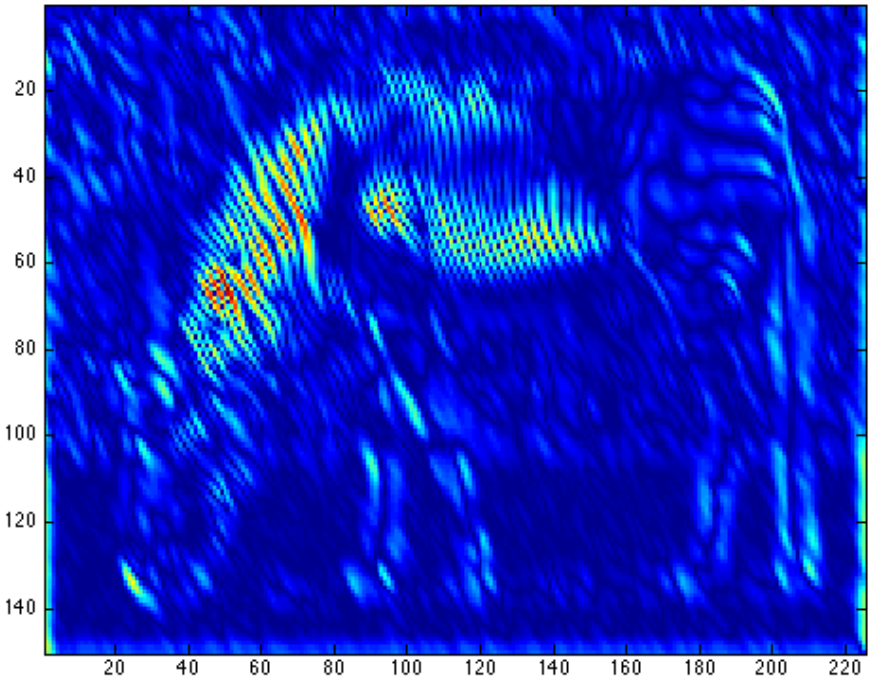


Slide adaptat după K. Grauman

Imagini filtrate cu răspuns absolut

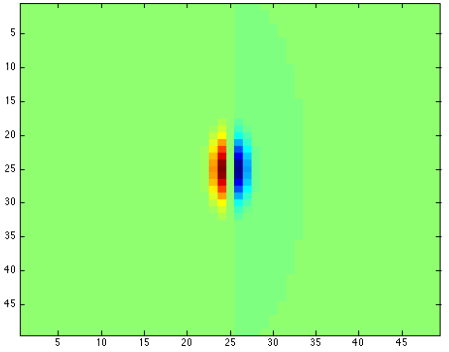






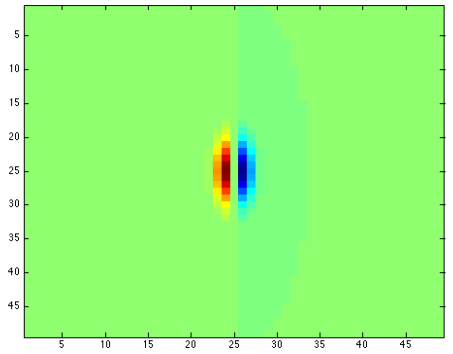
Slide adaptat după K. Grauman

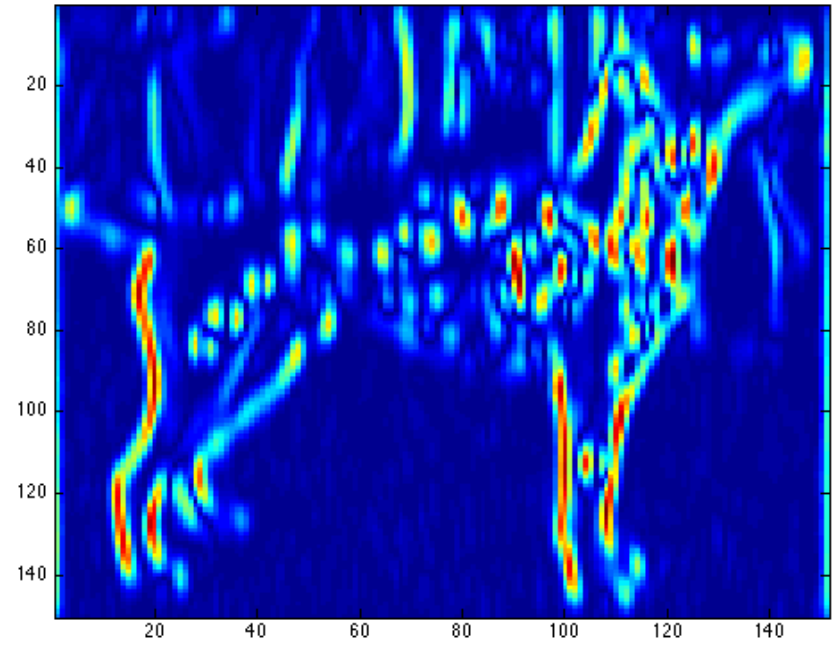
Imagini + filtru

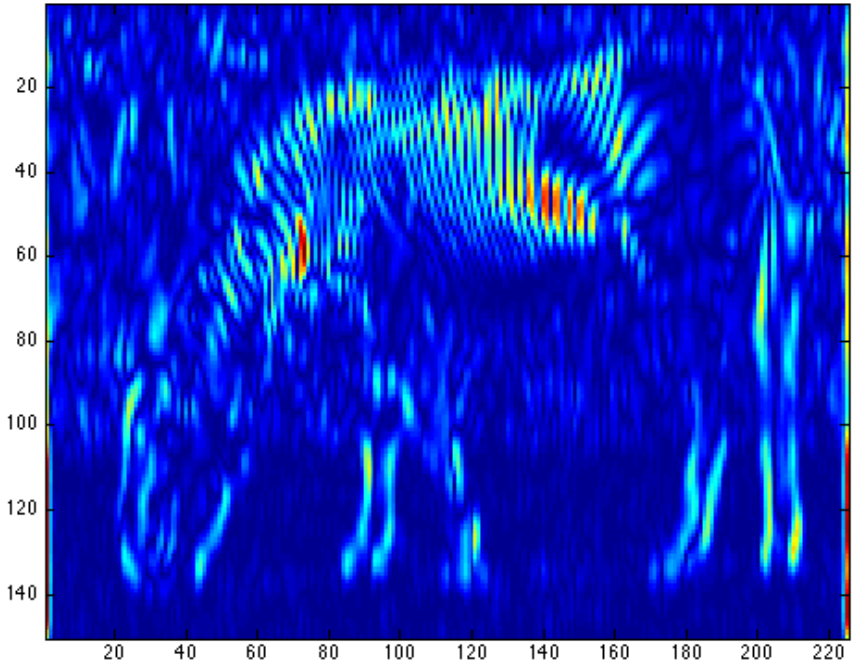


Slide adaptat după K. Grauman

Imagini filtrate cu răspuns absolut

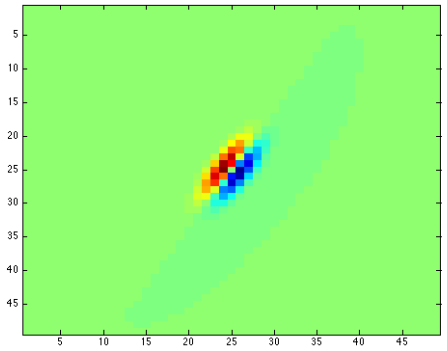






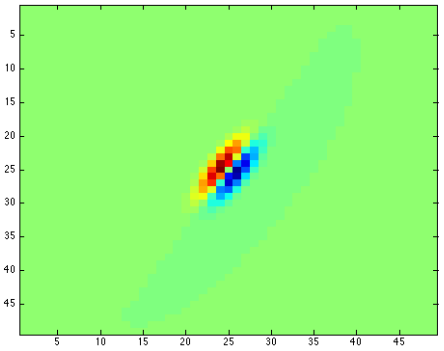
Slide adaptat după K. Grauman

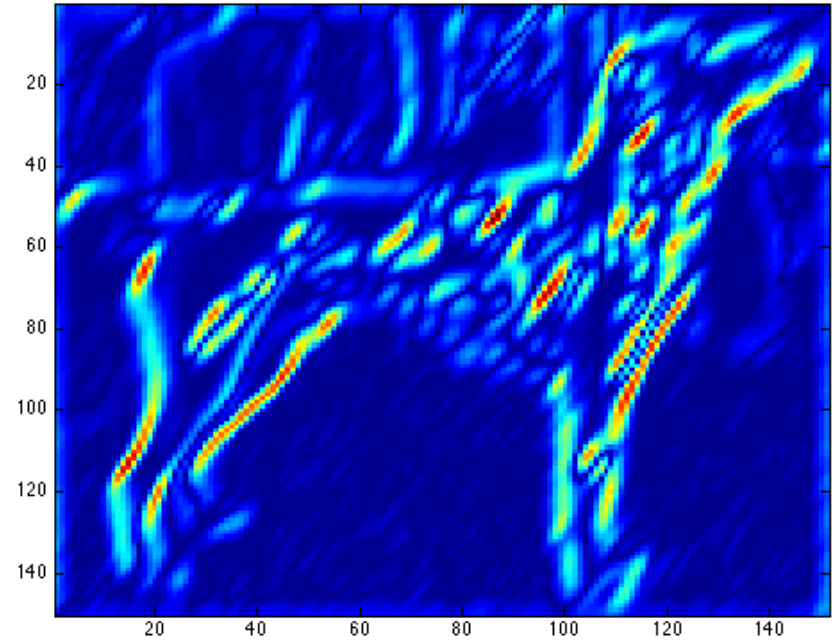
Imagini + filtru

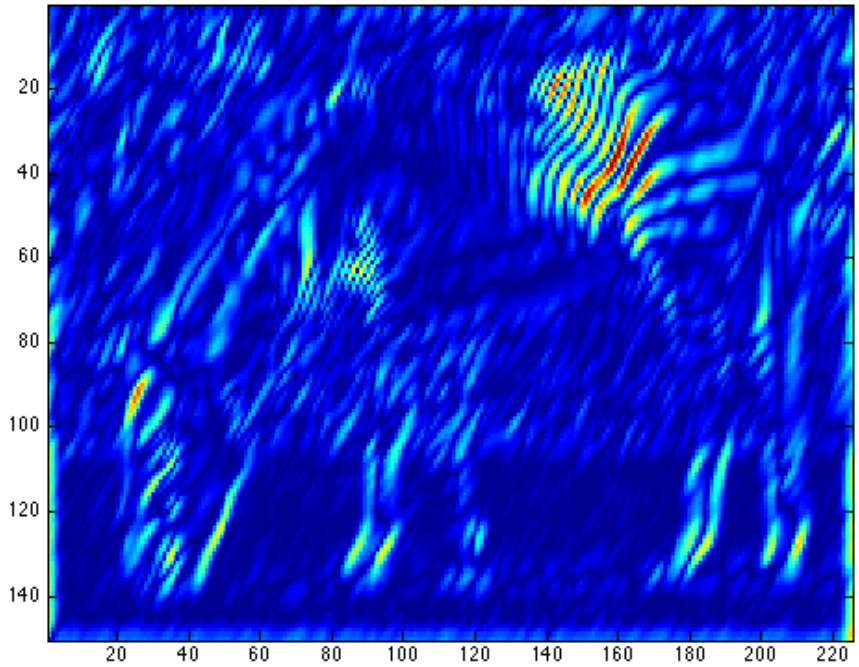


Slide adaptat după K. Grauman

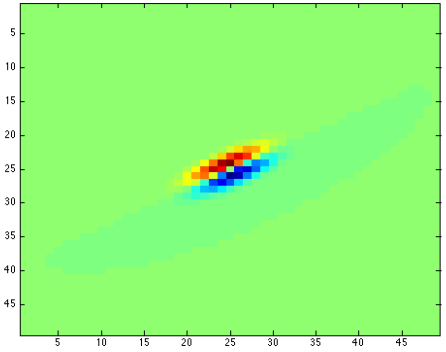
Imagini filtrate cu răspuns absolut





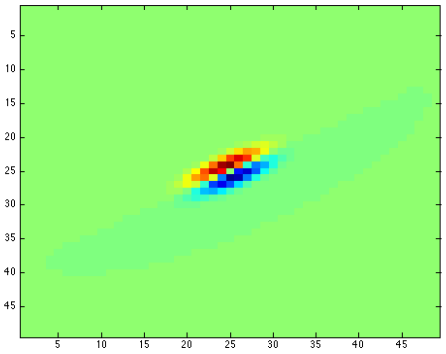


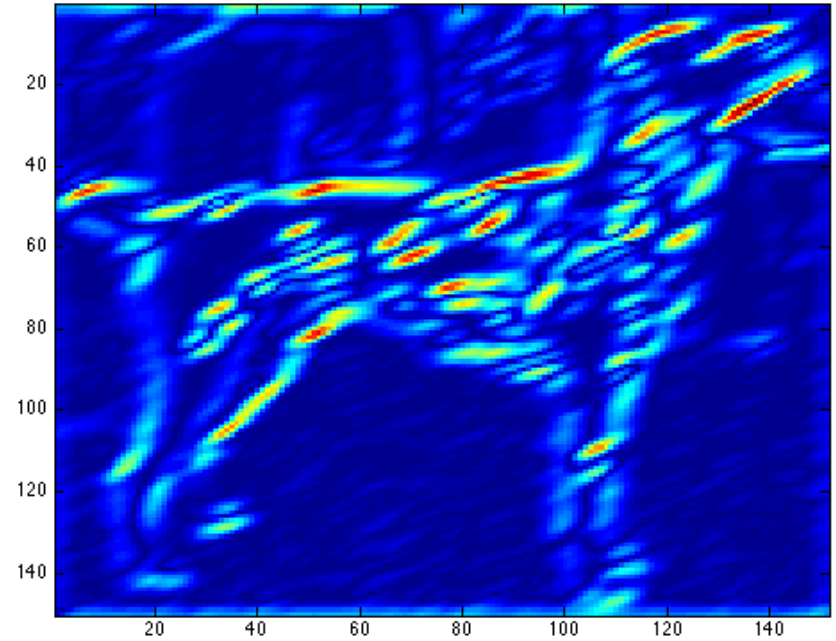
Imagini + filtru

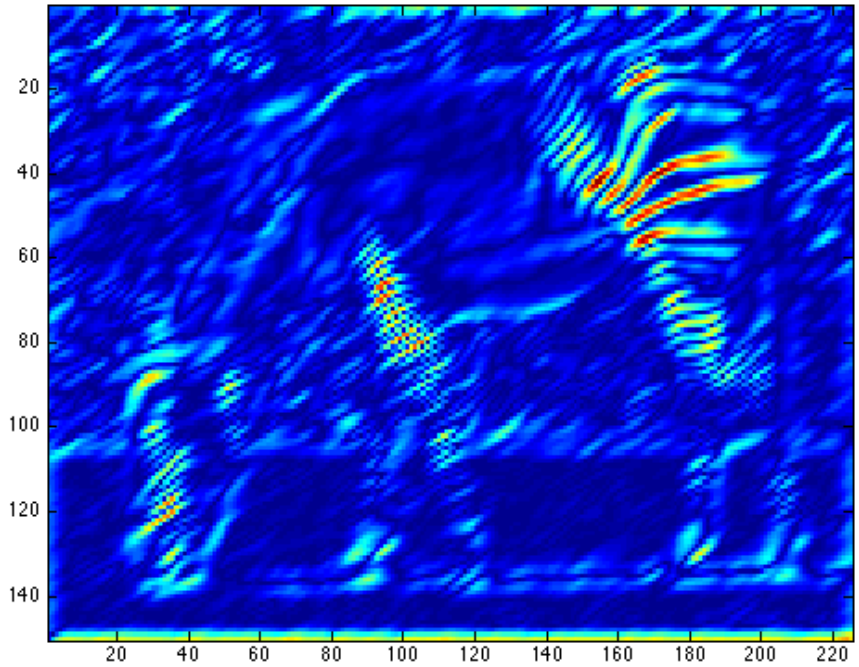


Slide adaptat după K. Grauman

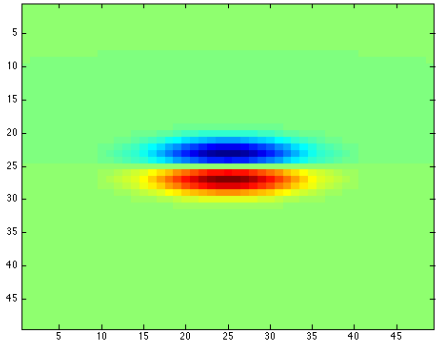
Imagini filtrate cu răspuns absolut





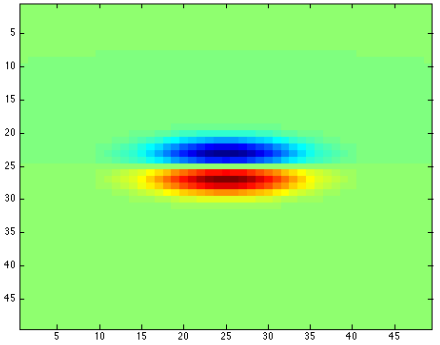


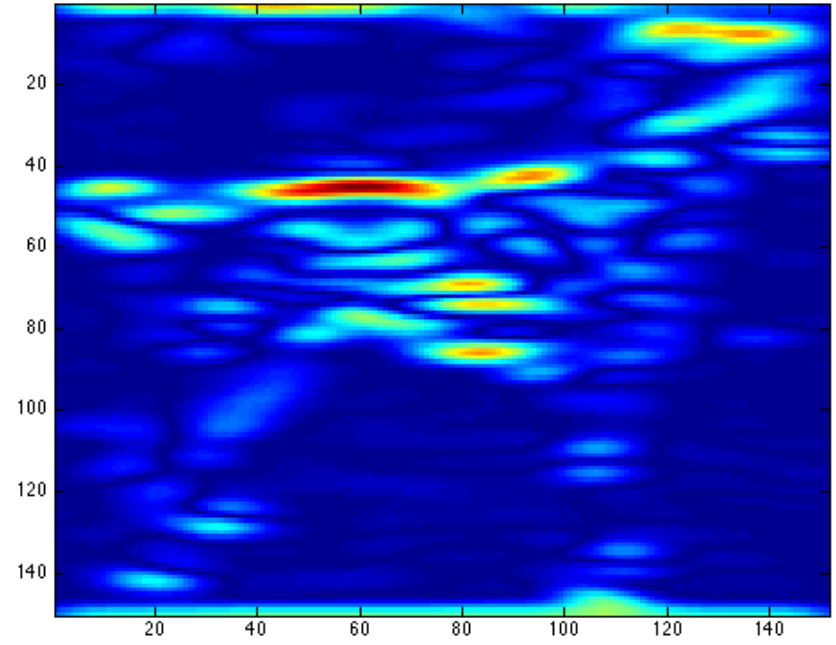
Imagini + filtru

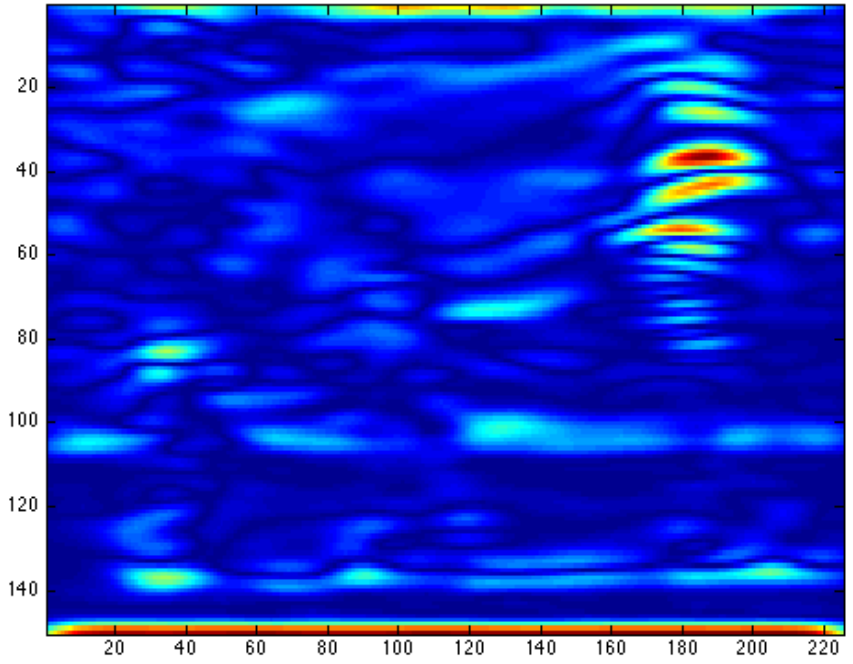


Slide adaptat după K. Grauman

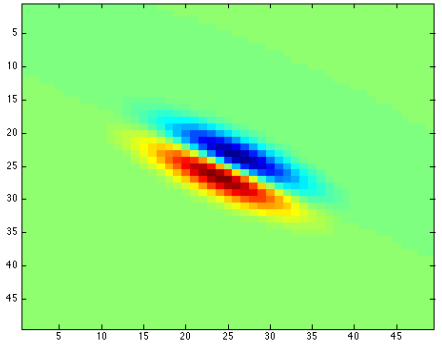
Imagini filtrate
cu răspuns absolut





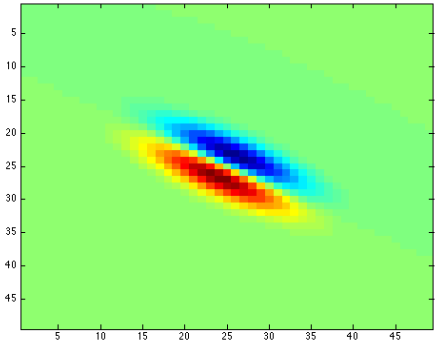


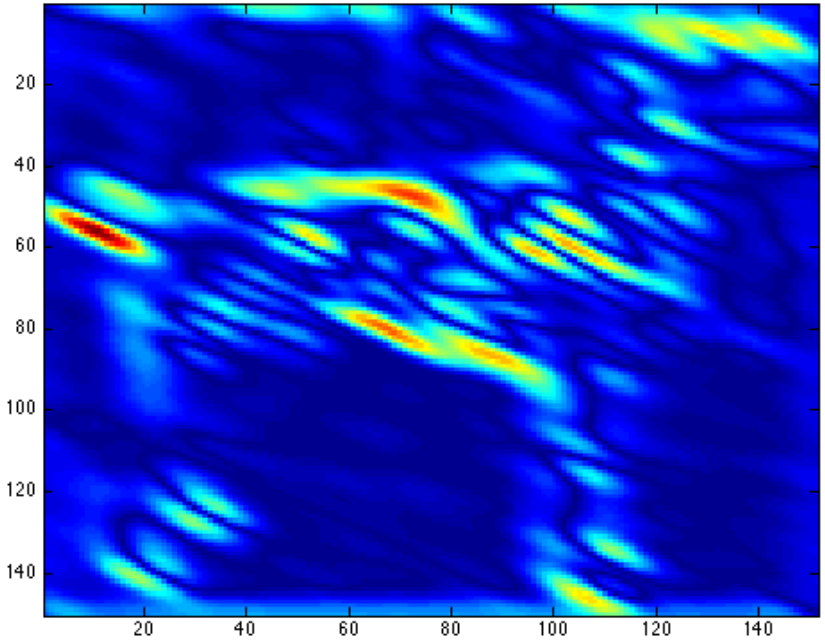
Imagini + filtru

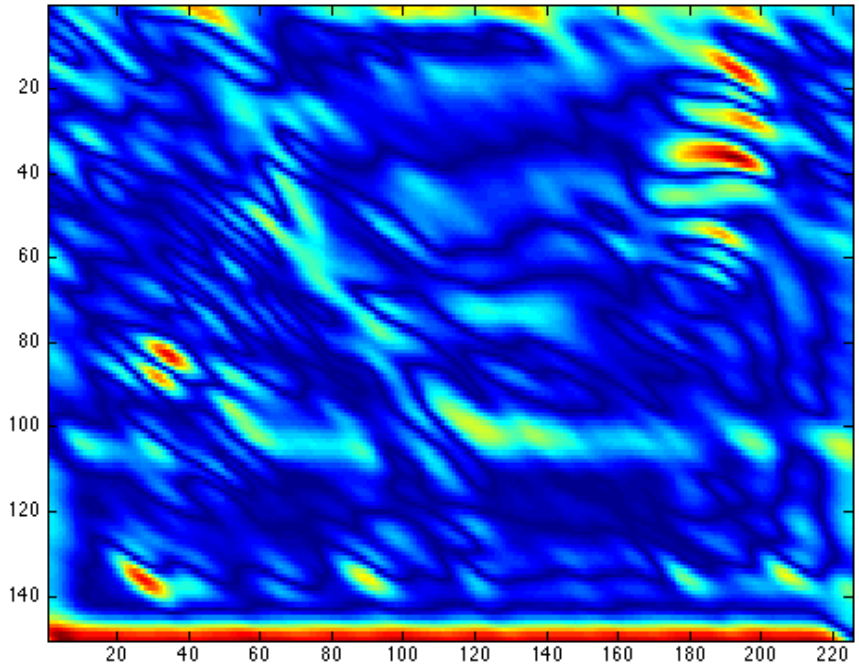


Slide adaptat după K. Grauman

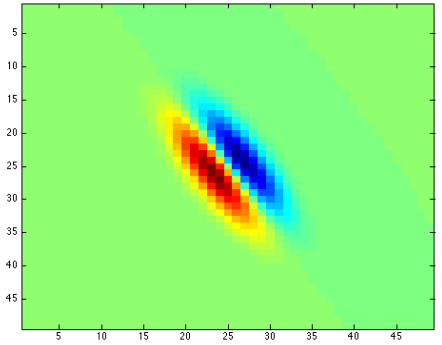
Imagini filtrate
cu răspuns absolut





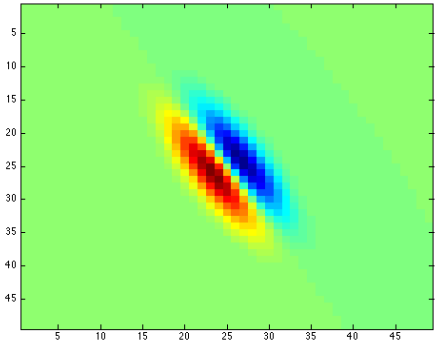


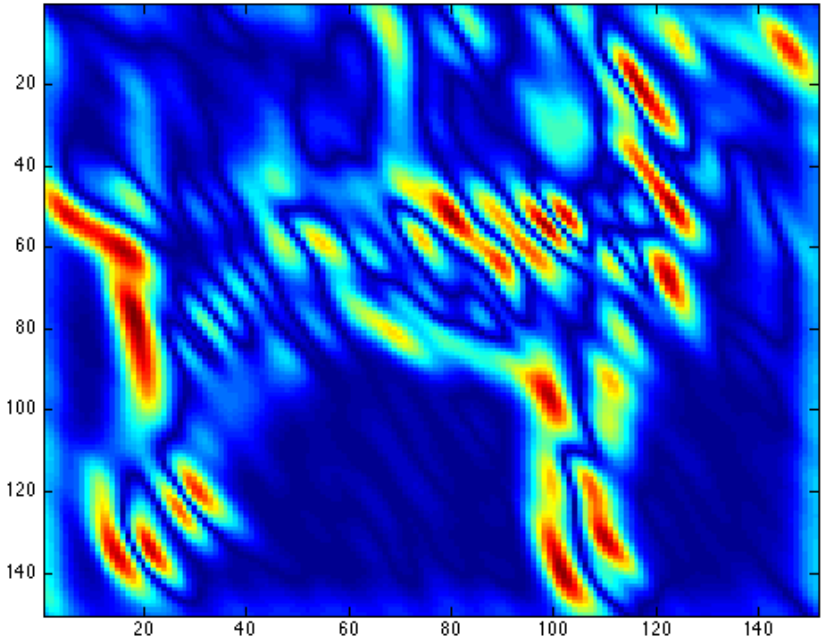
Imagini + filtru

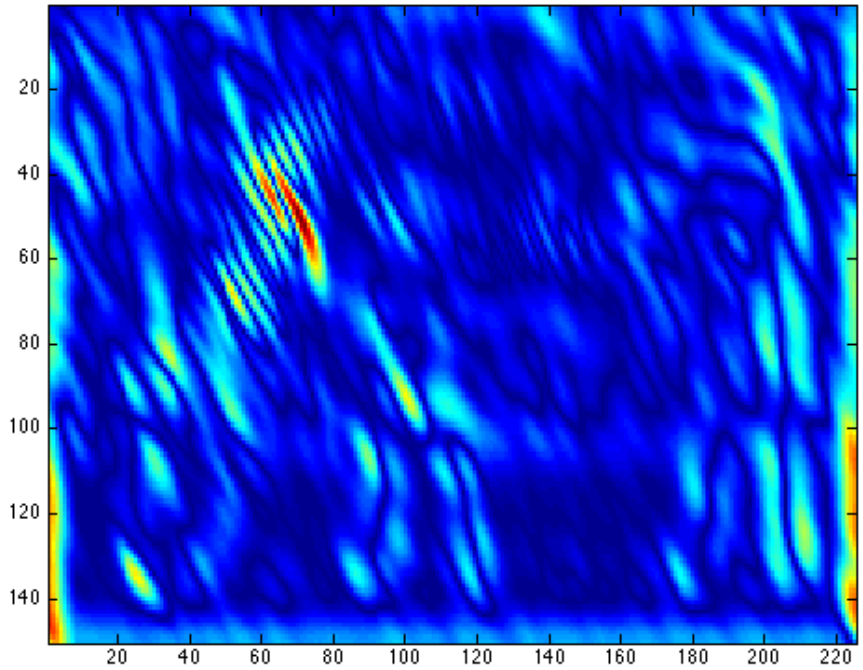


Slide adaptat după K. Grauman

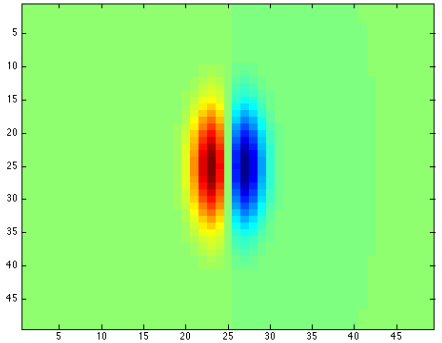
Imagini filtrate
cu răspuns absolut





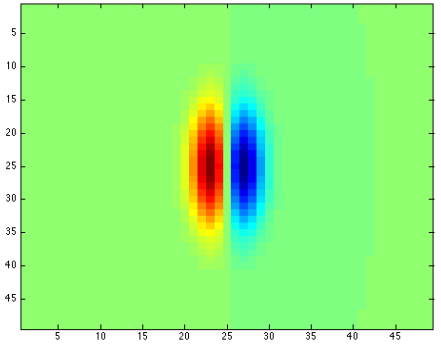


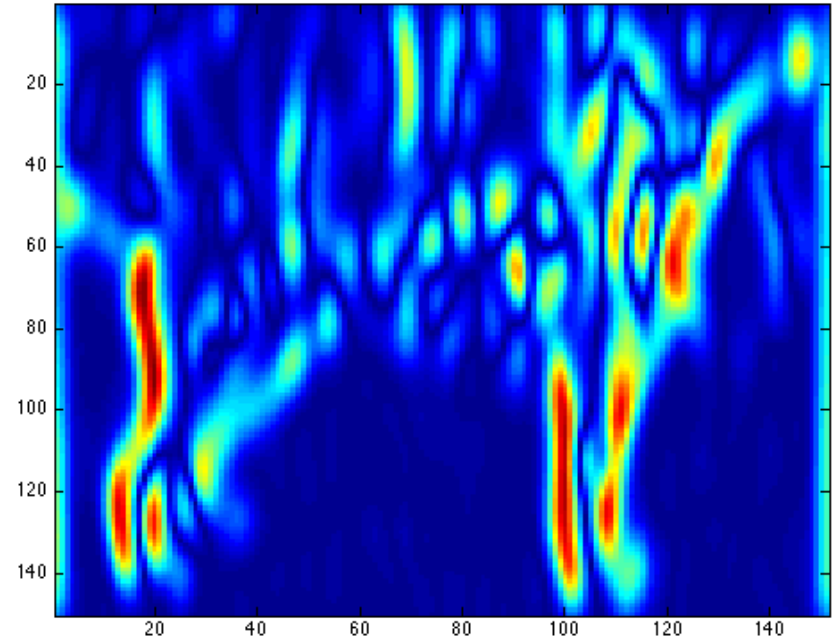
Imagini + filtru

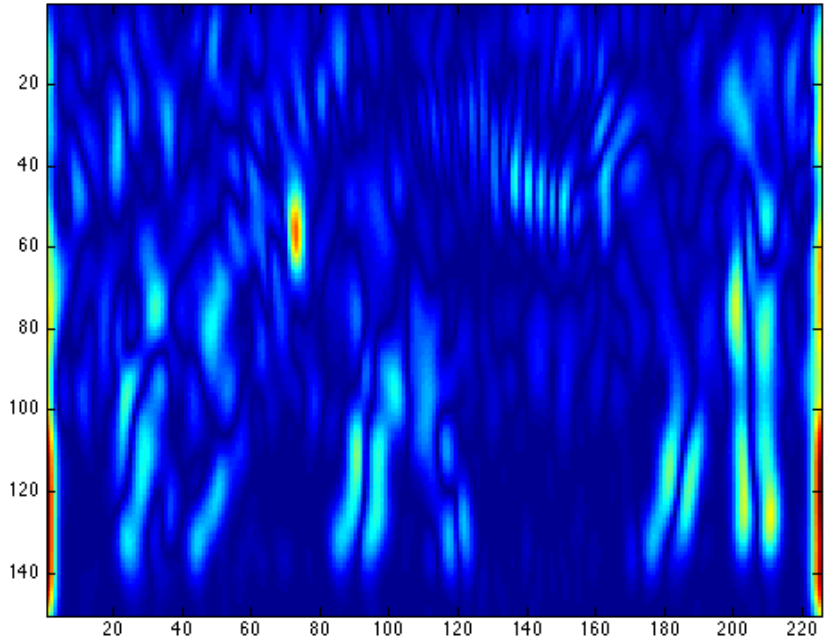


Slide adaptat după K. Grauman

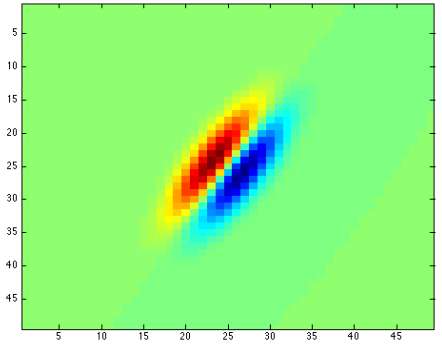
Imagini filtrate cu răspuns absolut





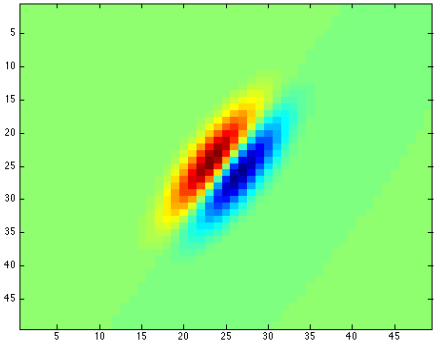


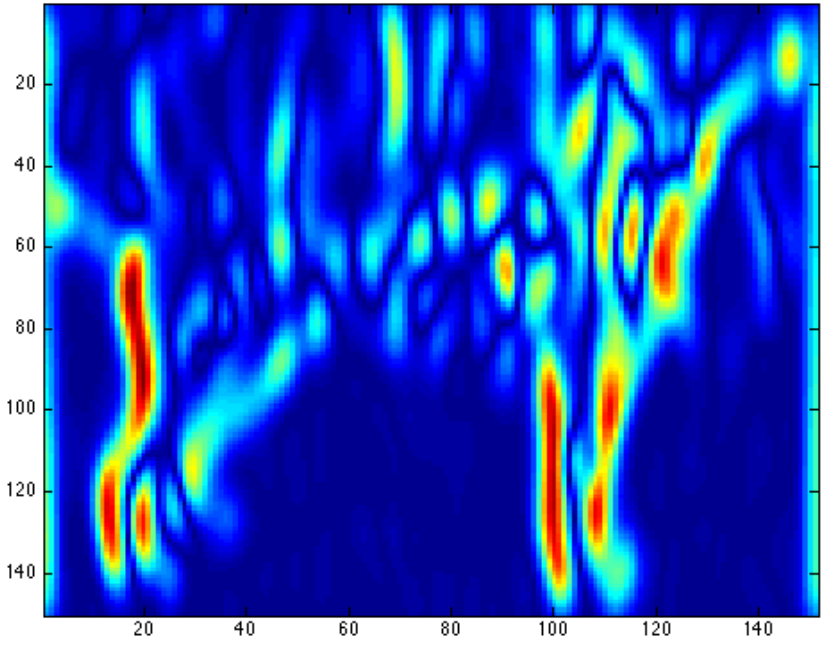
Imagini + filtru

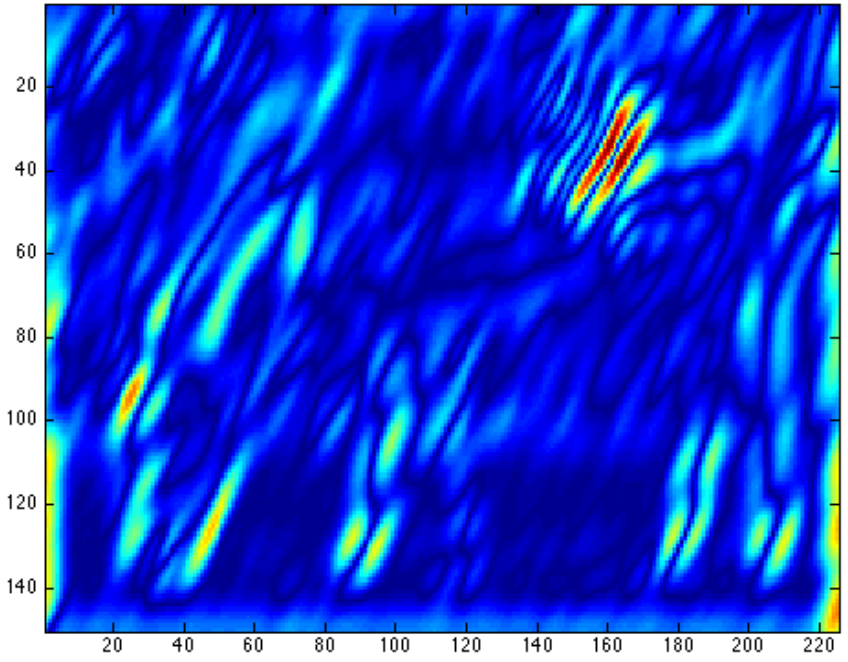


Slide adaptat după K. Grauman

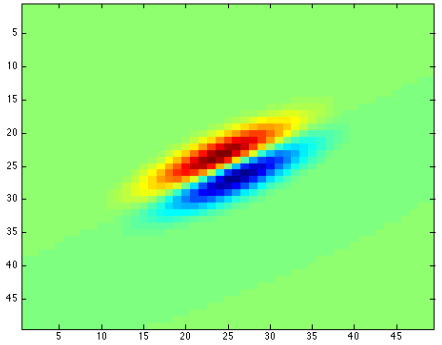
Imagini filtrate
cu răspuns absolut





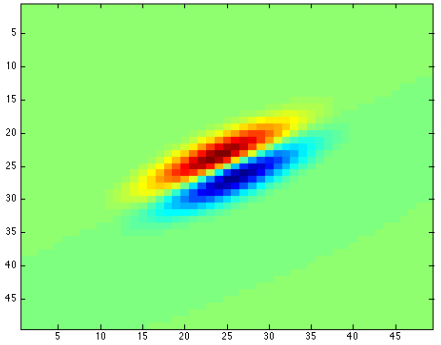


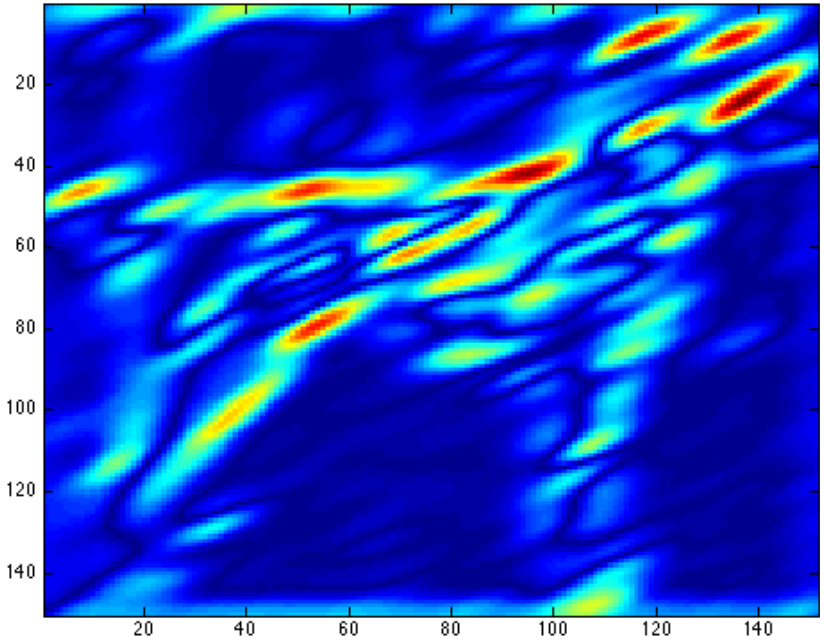
Imagini + filtru

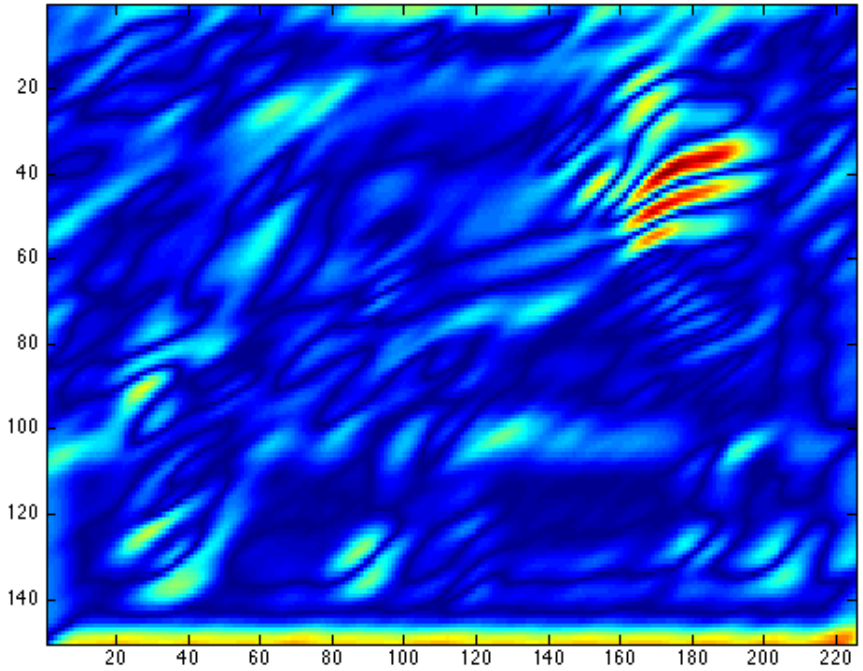


Slide adaptat după K. Grauman

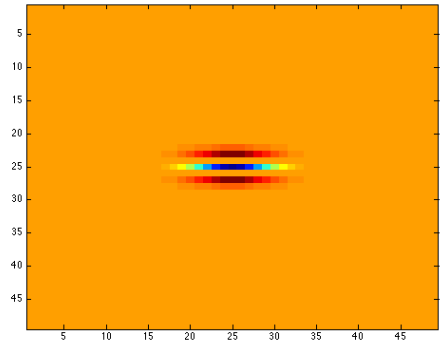
Imagini filtrate
cu răspuns absolut





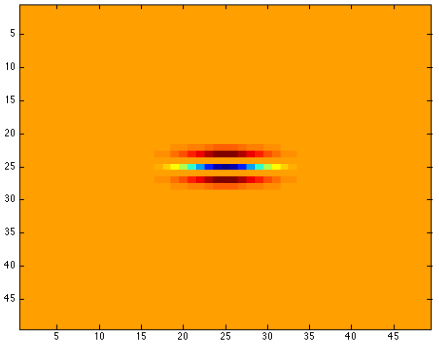


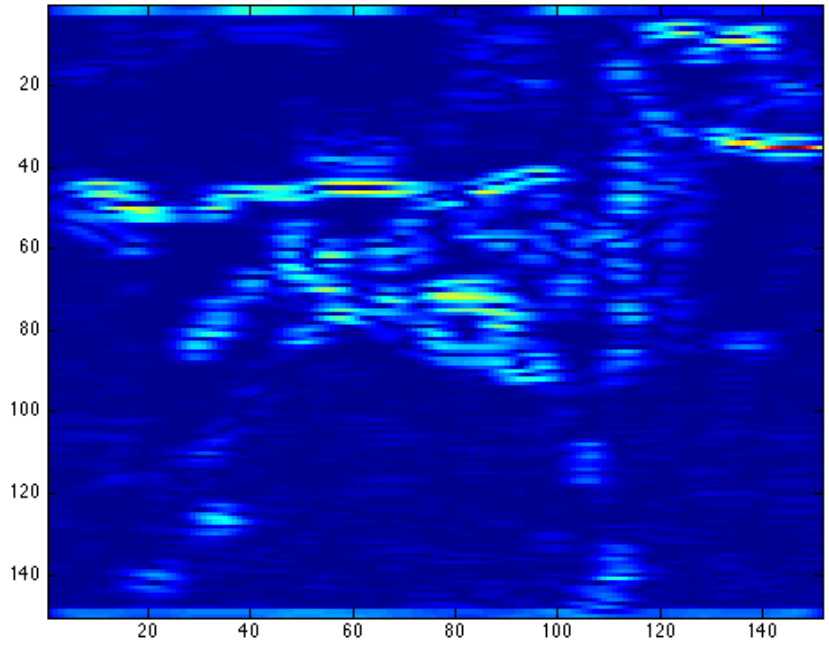
Imagini + filtru

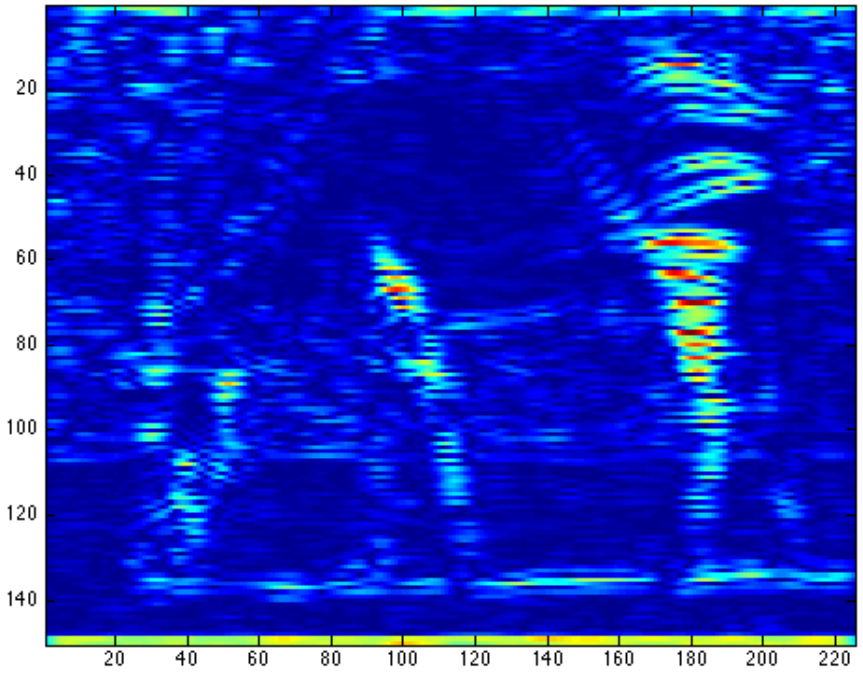


Slide adaptat după K. Grauman

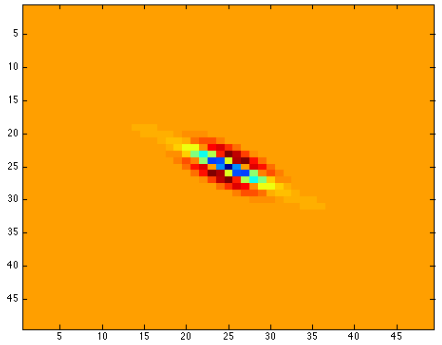
Imagini filtrate cu răspuns absolut





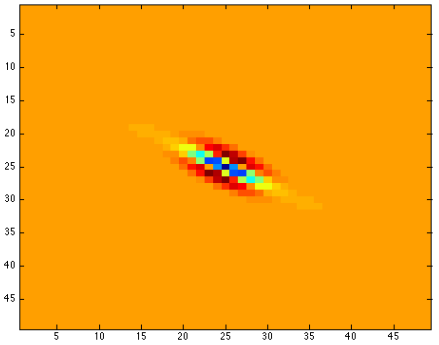


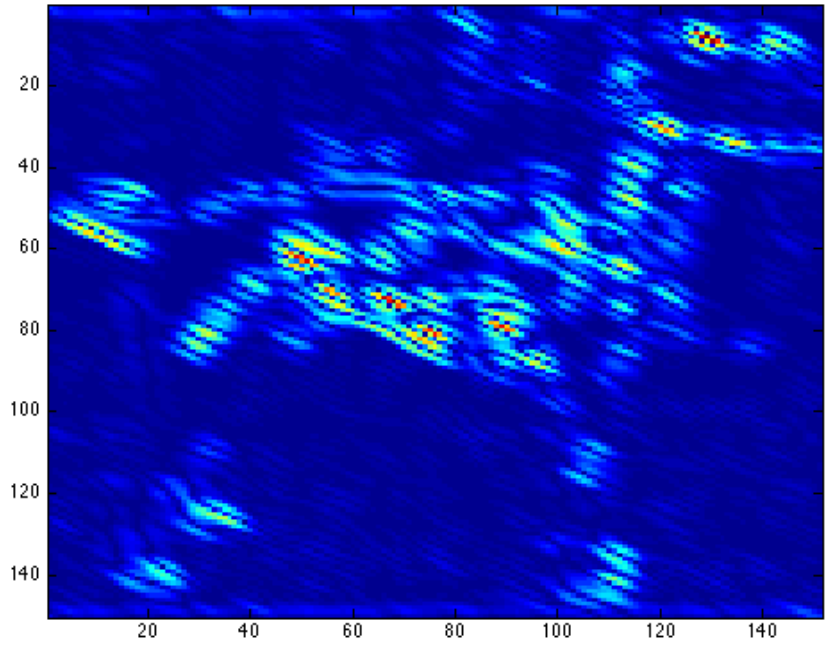
Imagini + filtru

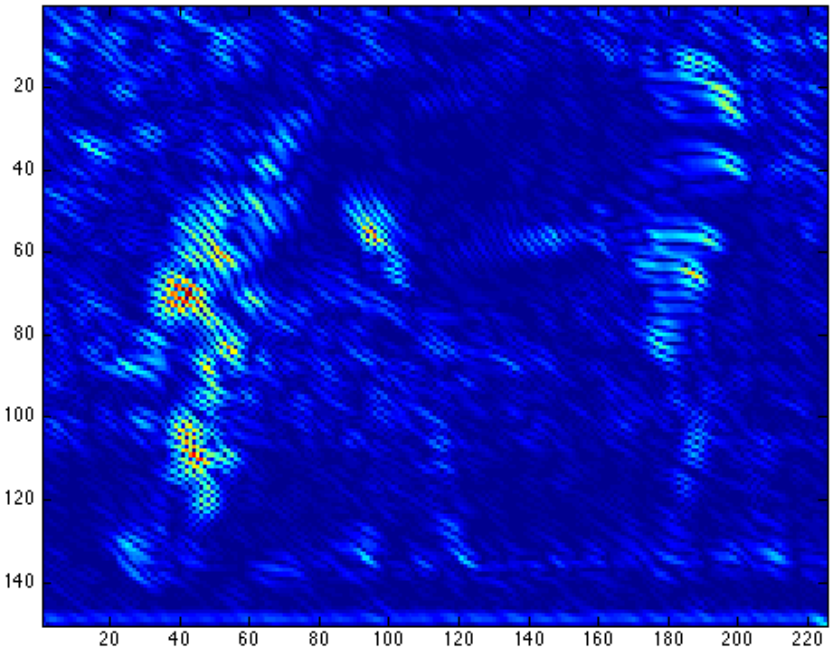


Slide adaptat după K. Grauman

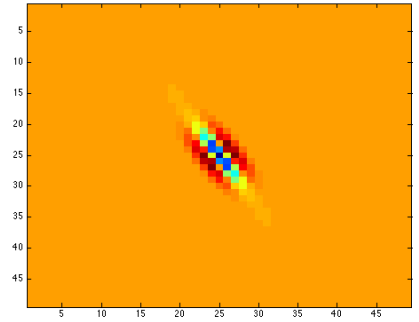
Imagini filtrate cu răspuns absolut





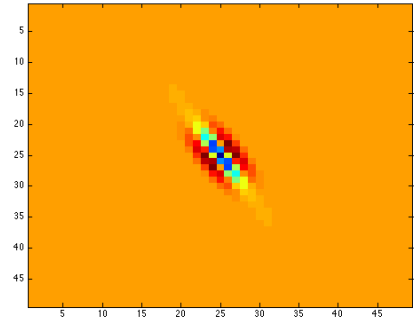


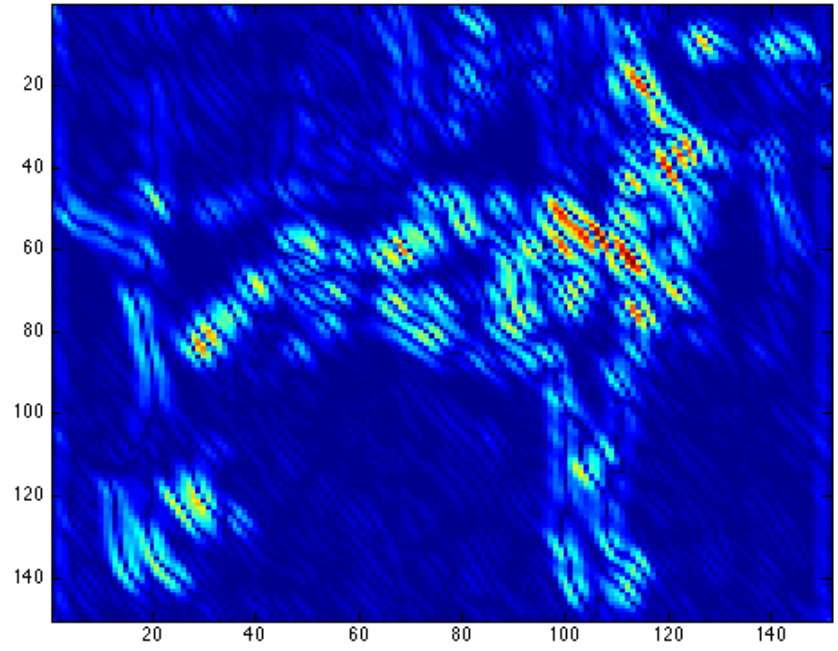
Imagini + filtru

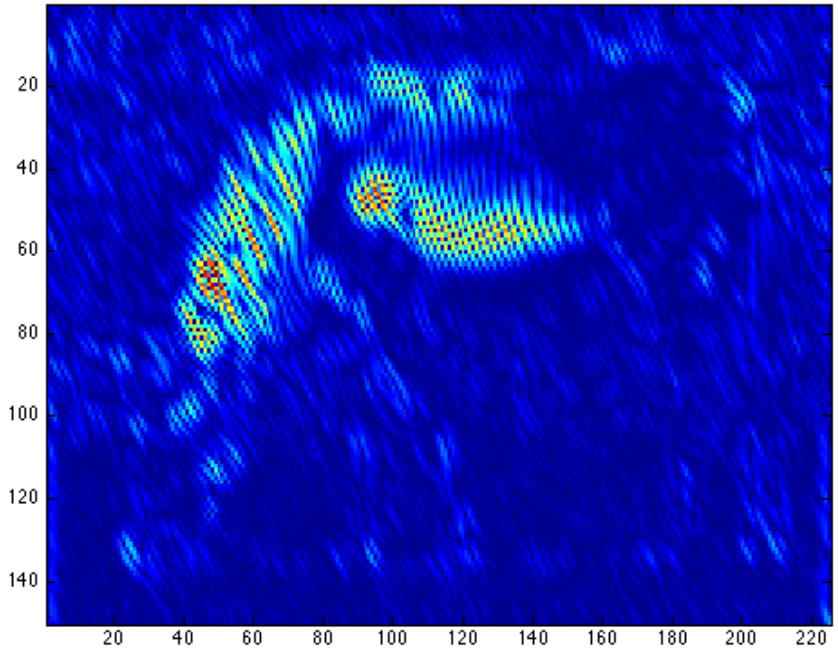


Slide adaptat după K. Grauman

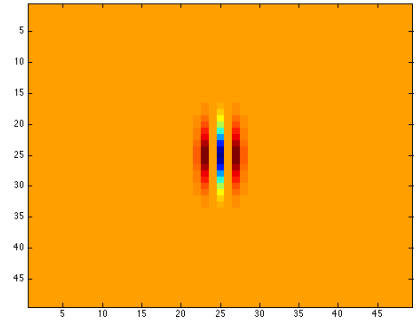
Imagini filtrate
cu răspuns absolut





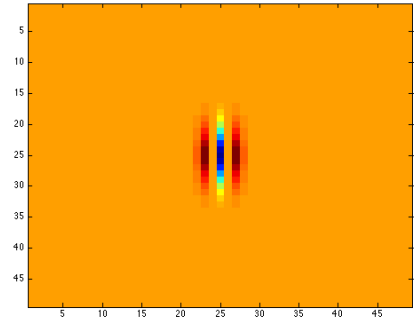


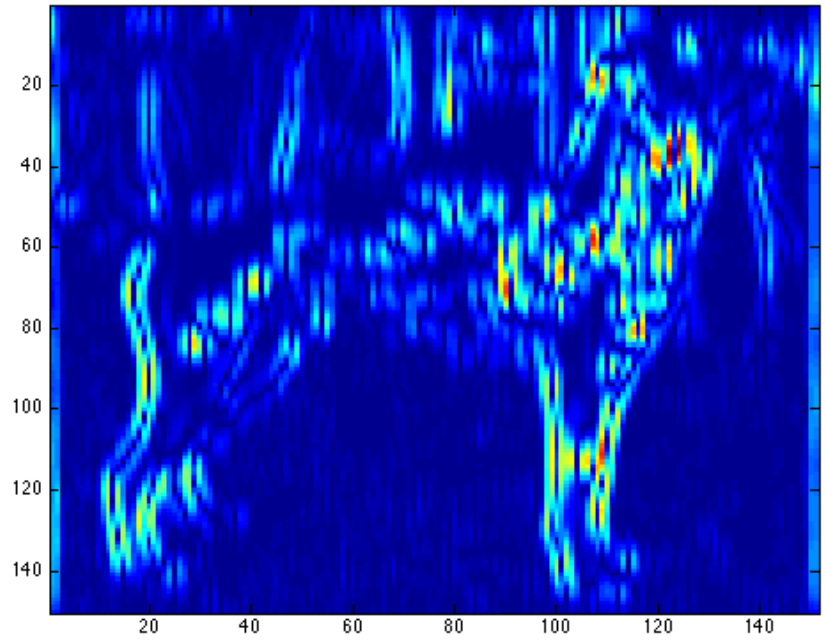
Imagini + filtru

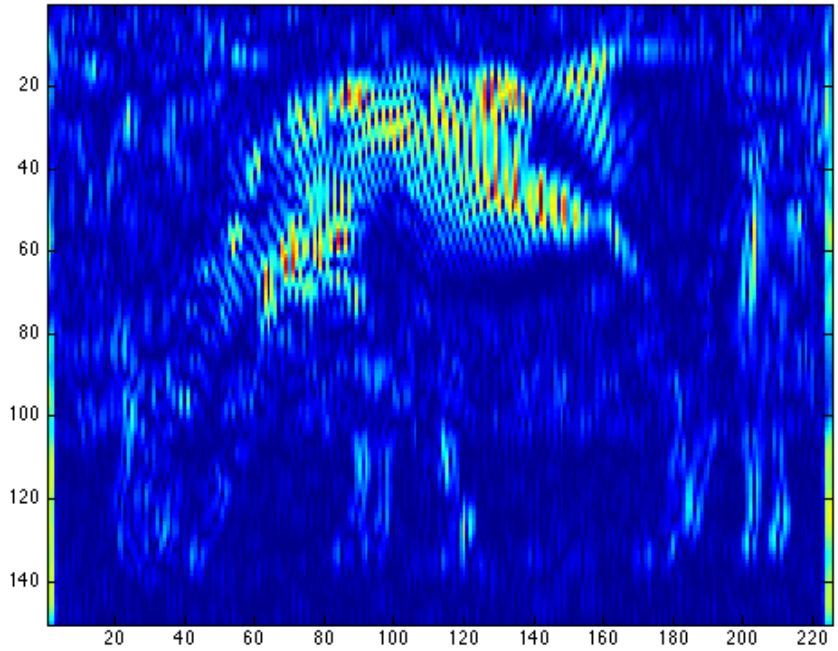


Slide adaptat după K. Grauman

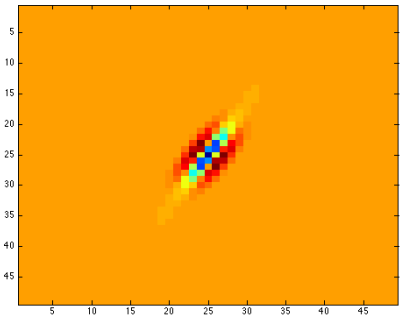
Imagini filtrate
cu răspuns absolut





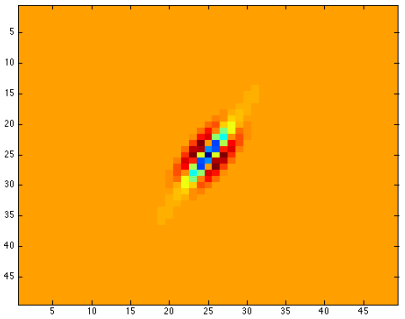


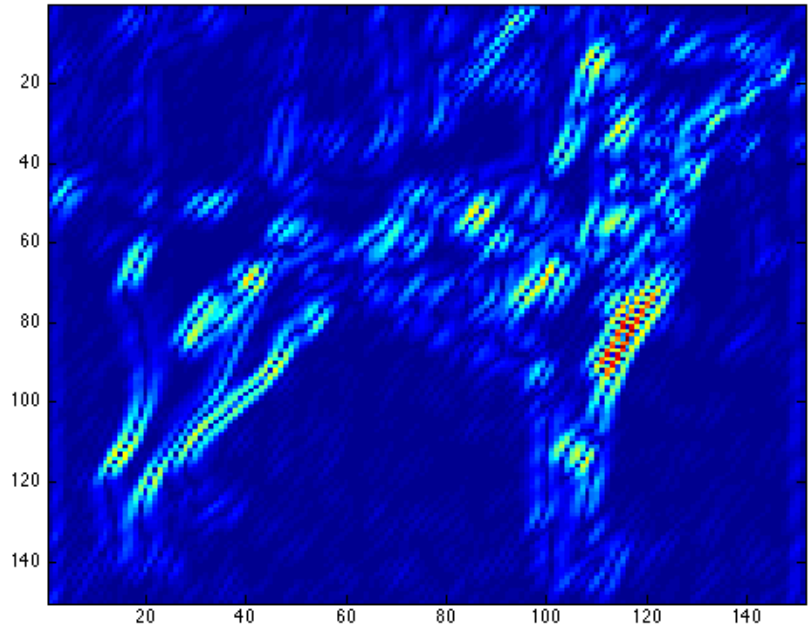
Imagini + filtru

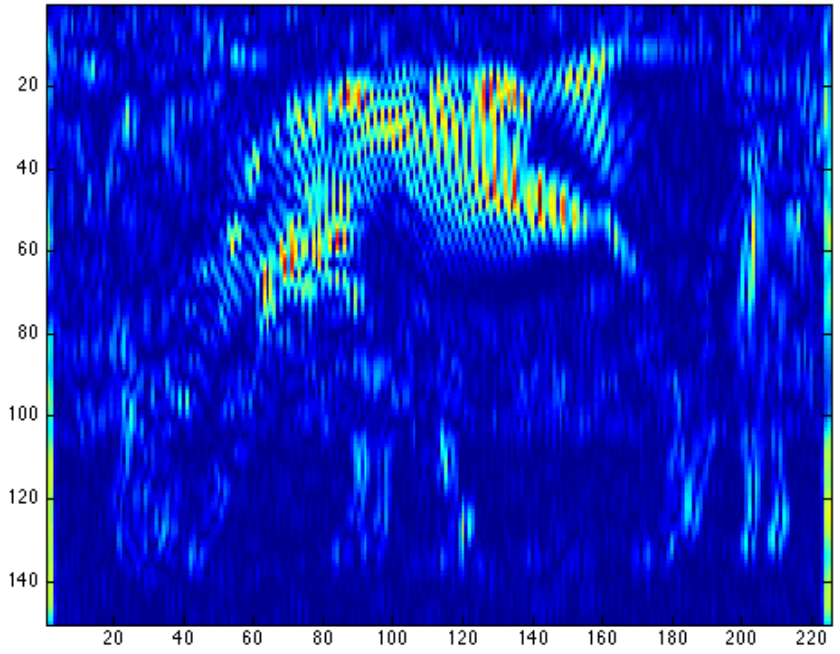


Slide adaptat după K. Grauman

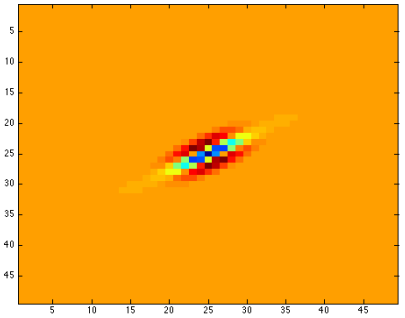
Imagini filtrate
cu răspuns absolut





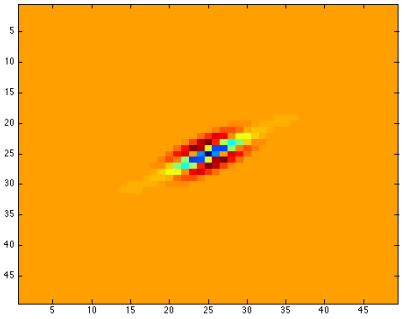


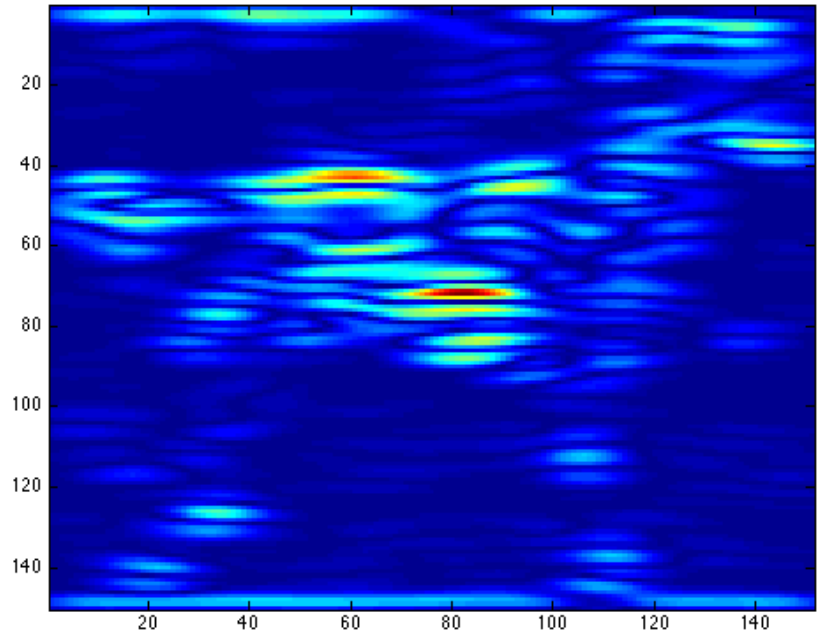
Imagini + filtru

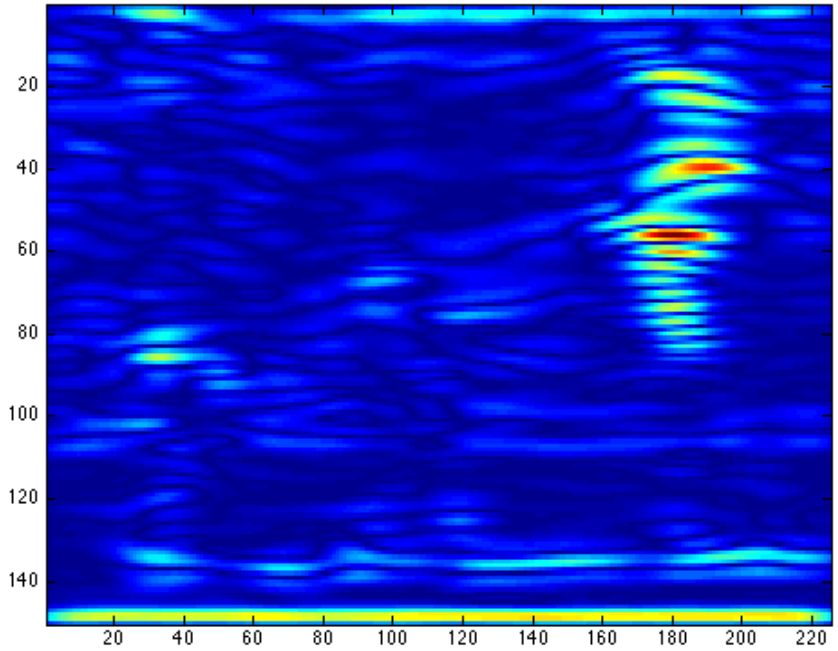


Slide adaptat după K. Grauman

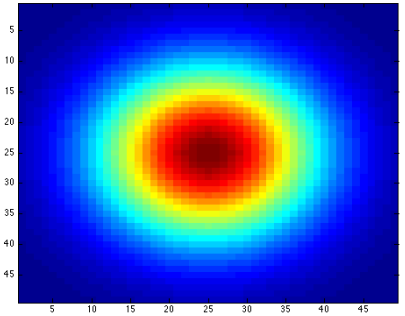
Imagini filtrate
cu răspuns absolut





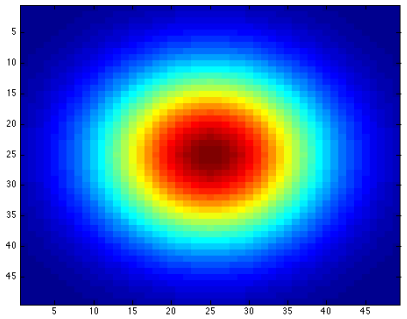


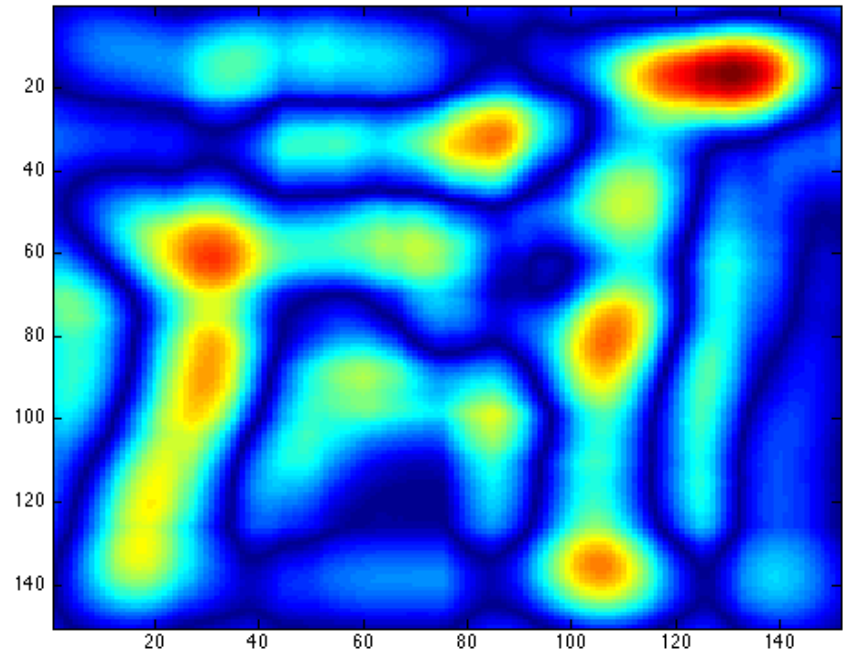
Imagini + filtru

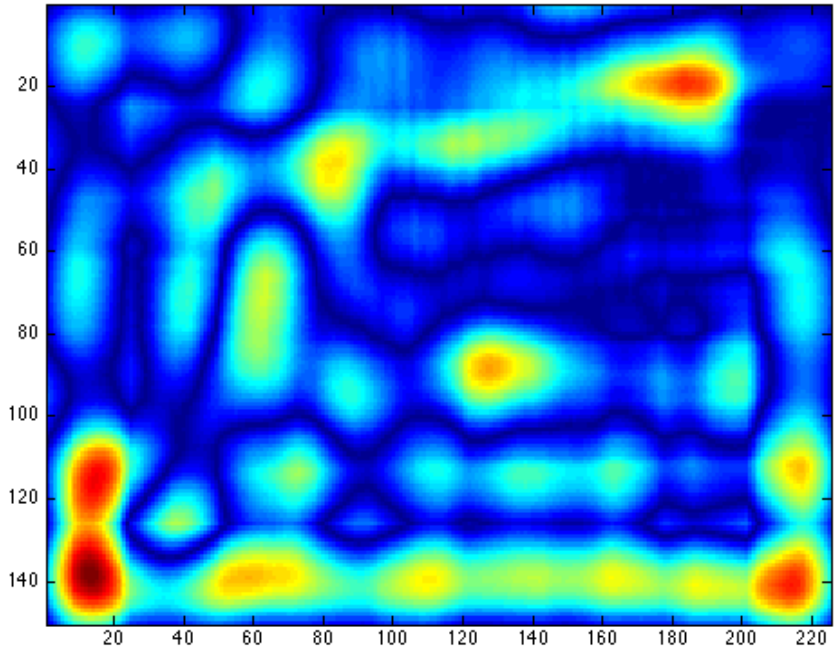


Slide adaptat după K. Grauman

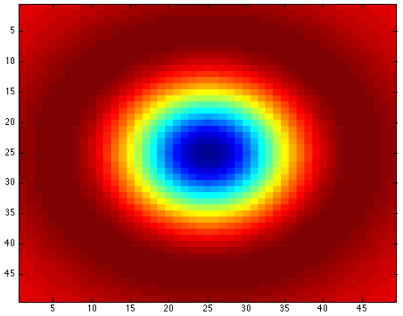
Imagini filtrate
cu răspuns absolut





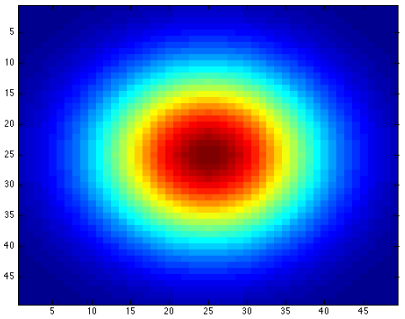


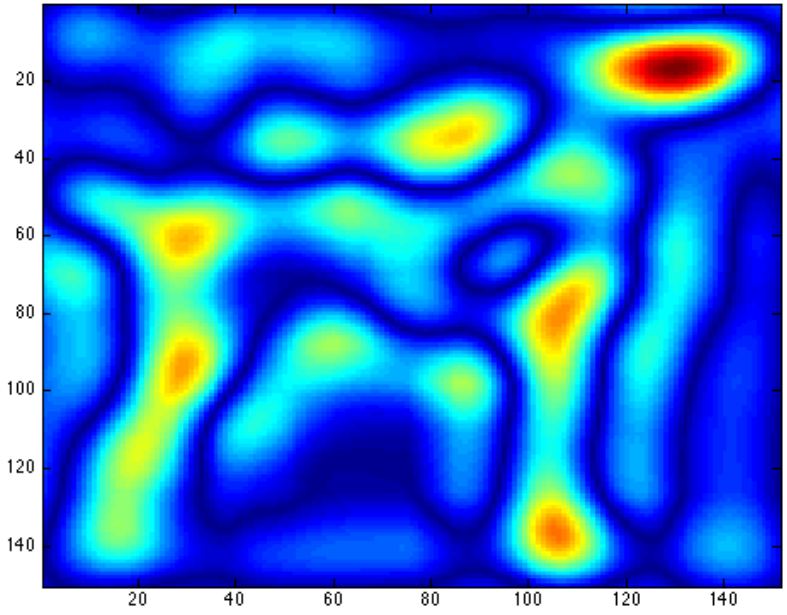
Imagini + filtru

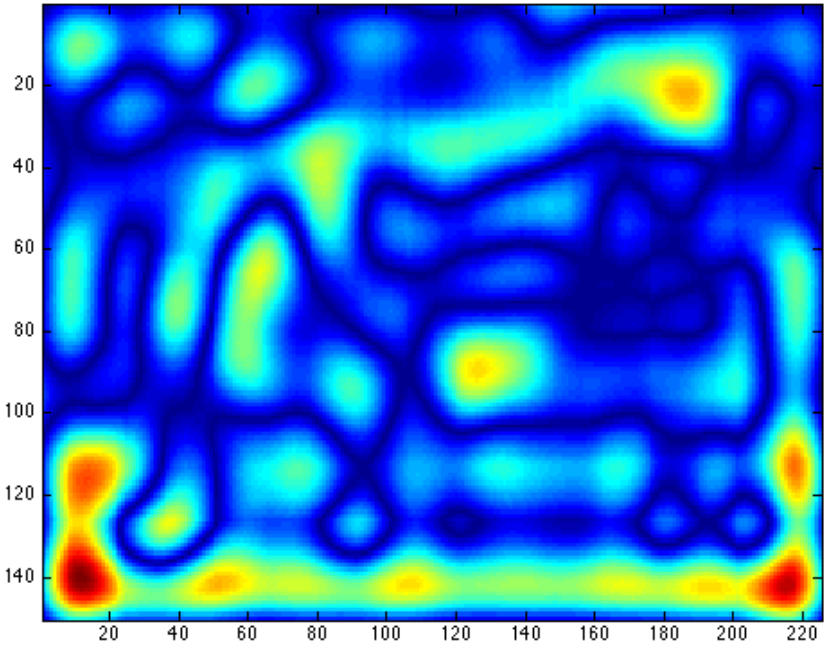


Slide adaptat după K. Grauman

Imagini filtrate
cu răspuns absolut

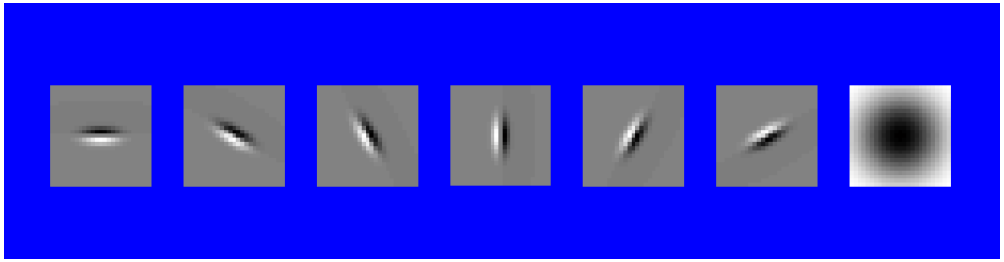




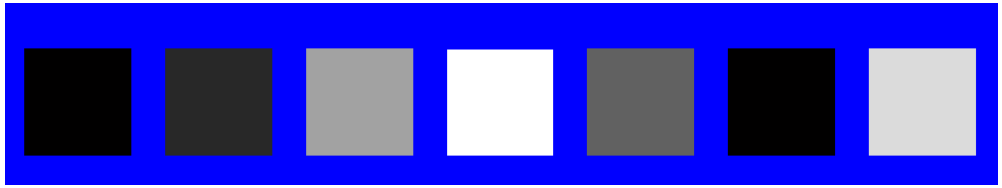


Găsiți corespondența!

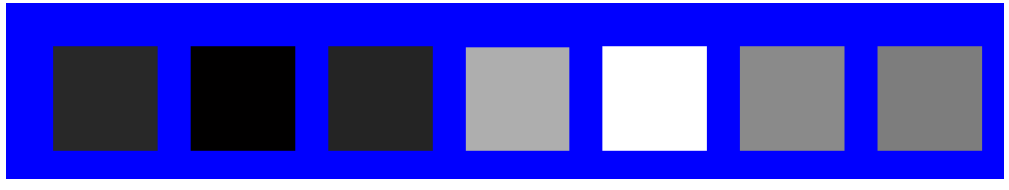
Colecție de filtre



1



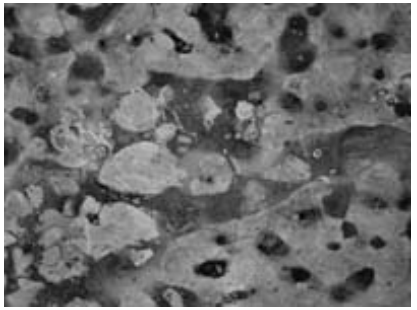
2



3

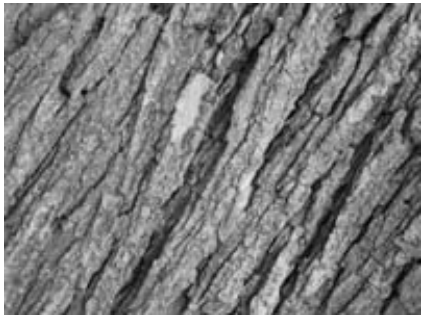
Răspunsul mediu – valoare absolută (modul)

A



B

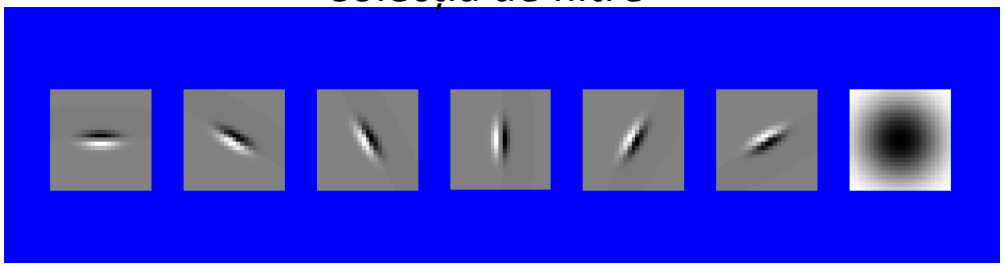
C



Slide adaptat după D. Hoiem

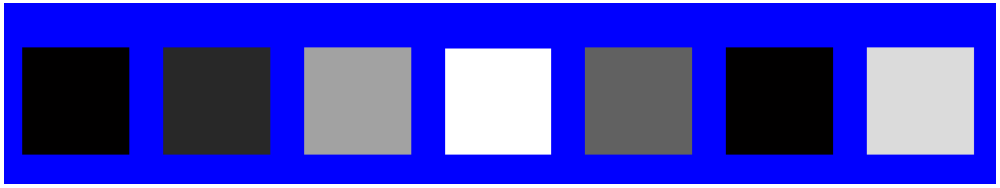
Reprezentarea texturilor pe baza răspunsului mediu în modul

Colecția de filtre



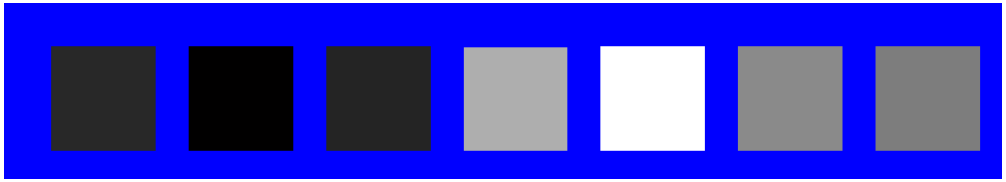
B

1

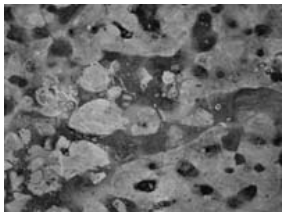


C

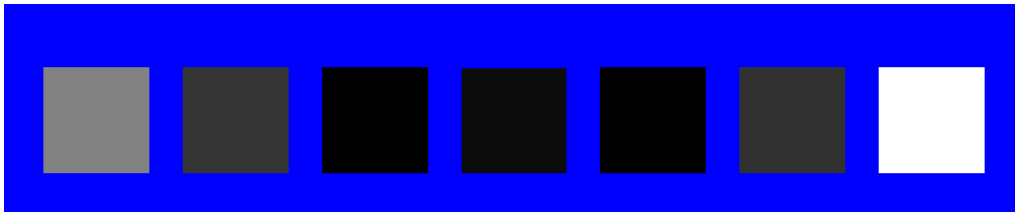
2



A



3



Răspunsul mediu – valoare absolută (modul)

Slide adaptat după D. Hoiem

De ce studiem textura?

- **Segmentare/clasificare**
  - analiza, reprezentarea texturii
  - grupează regiunile din imagine cu aceeaşi textură

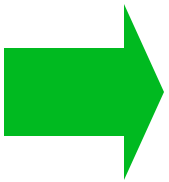
- **Sinteză**
  - dată o mostră de textură (dimensiuni mici) vrem să generăm o textură similară (dimensiuni mari)

mostră de textură

textură generată

Sinteza texturii

- Scop: generarea de noi exemple de textură (dimensiuni mari) pe baza unui mostre (dimensiuni mici)
- Aplicații: îmbogățirea mediilor virtuale cu textură, jocuri pe calculator, acoperirea găurilor



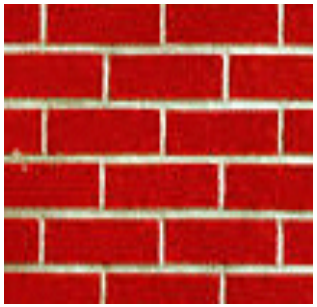
Slide adaptat după K. Grauman

Obstacole

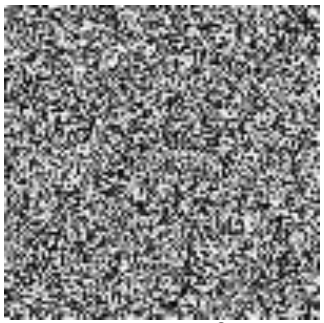
- Analiza texturii: cum cuprindem esența texturii?

- Texturi regulate – texeli care se repetă
- Texturi stocastice – nu prezintă în mod explicit texeli

- Vrem să modelăm întreg spectrul de texturi:
  de la texturi ce se repetă la texturi stocastice



repetare

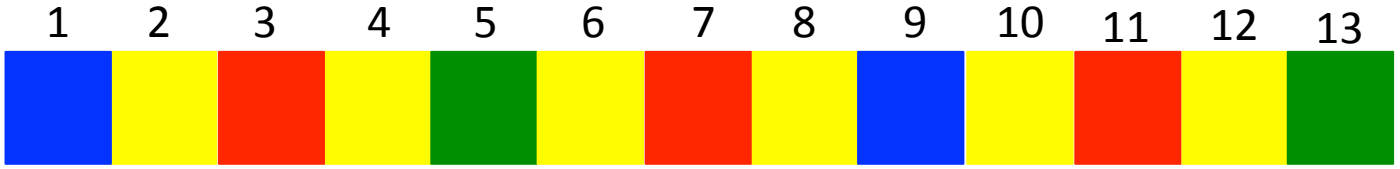


stocastic

amândouă

Slide adaptat după K. Grauman

Probabilitate



P(pixel =) = 2/13

P(pixel =) = 3/13

P(pixel =) = 6/13

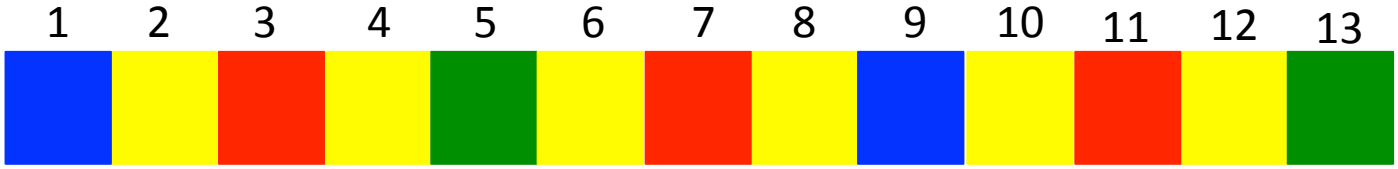
P(pixel =) = 2/13

2/13 + 6/13 + 3/13 + 2/13 = 1

Probabilitate = Nr cazuri favorabile / Nr cazuri totale

Suma = 1

Probabilitate condiționată



P(pixel = | pixelul din stânga =) = 3/6

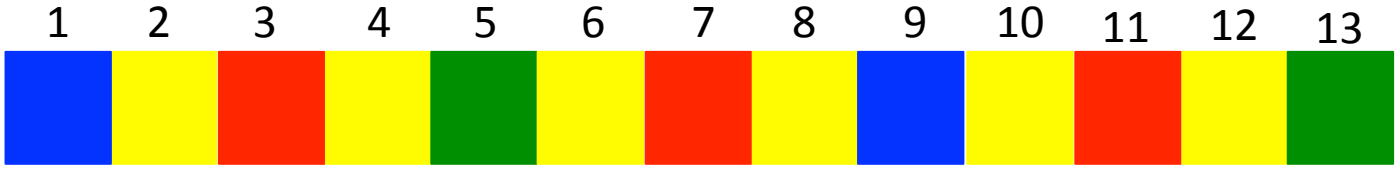
P(pixel = | pixelul din stânga =) = 1/6

P(pixel = | pixelul din stânga =) = 0/6

P(pixel = | pixelul din stânga =) = 2/6

3/6 + 1/6 + 0/6 + 2/6= 1

Probabilitate condiționată



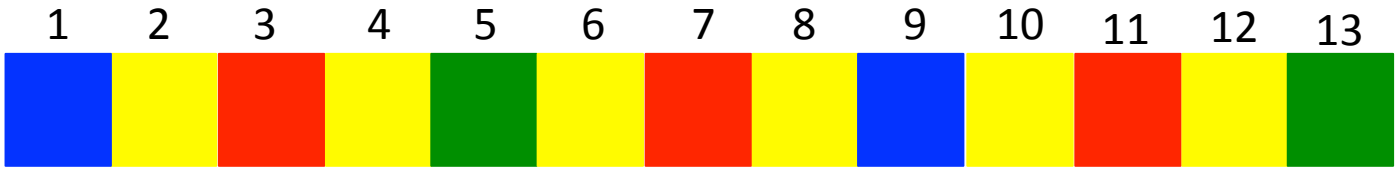
P(pixel =  | pixelul din stânga = ) = 0/2

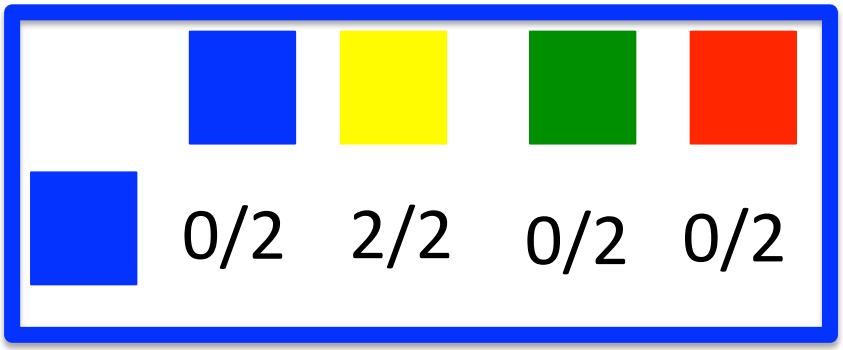
P(pixel =  | pixelul din stânga = ) = 0/2

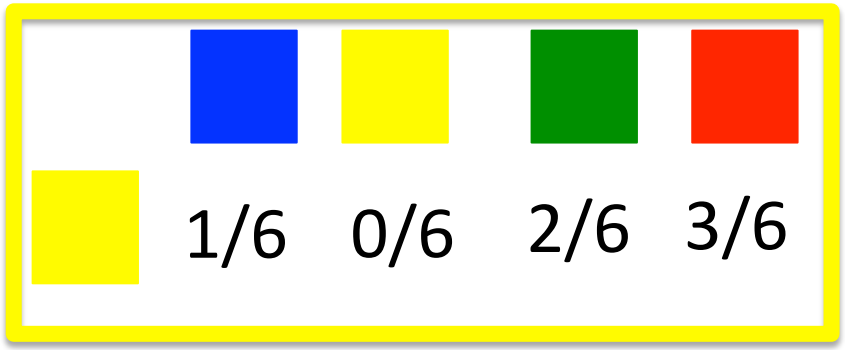
P(pixel = | pixelul din stânga = ) = 2/2

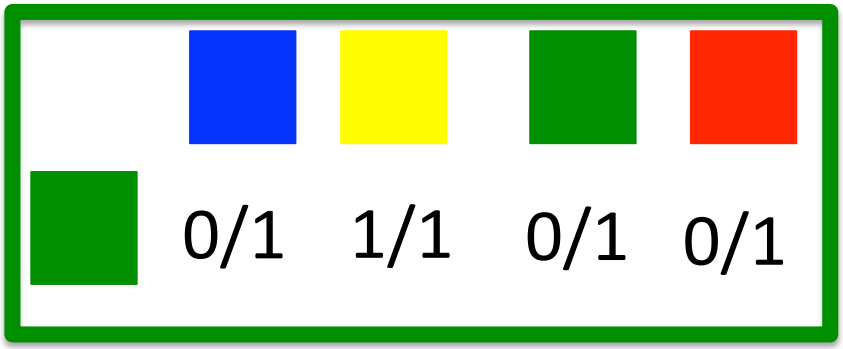
P(pixel =  | pixelul din stânga = ) = 0/2

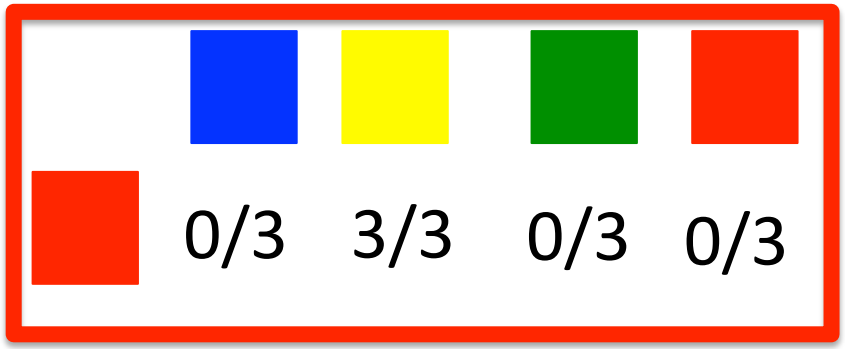
Probabilitate condiționată





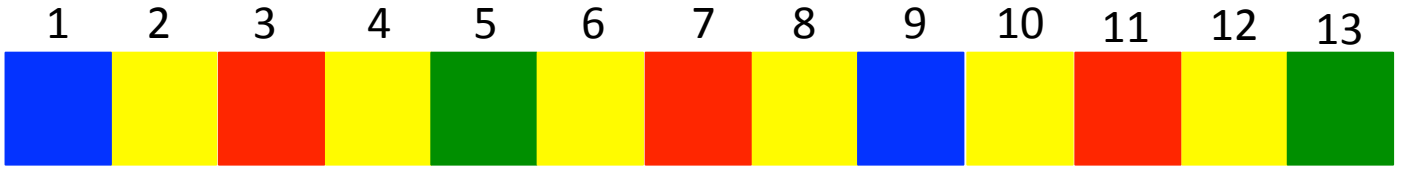


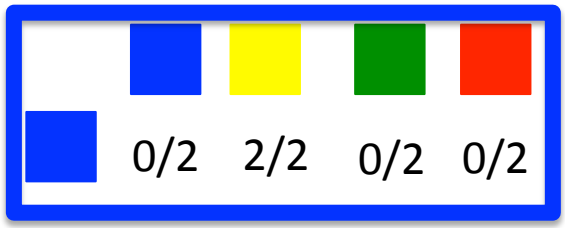


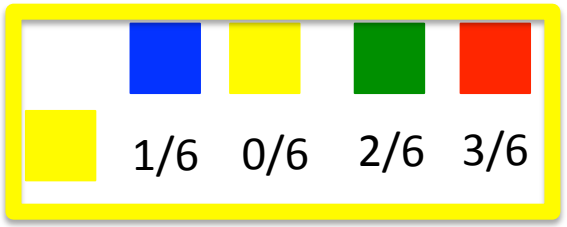


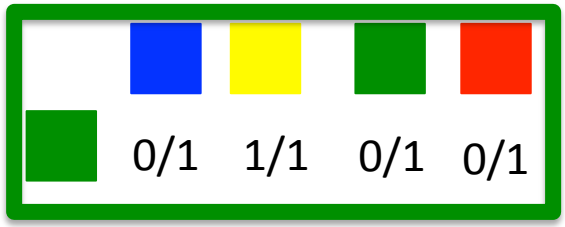
Completăm textura

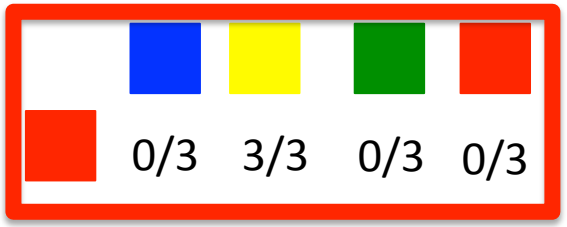
Completăm textura pixel cu pixel după următoarea regulă: observăm vecinul din stânga si apoi selectăm un pixel din distribuția condiționată (fără să o updatăm).





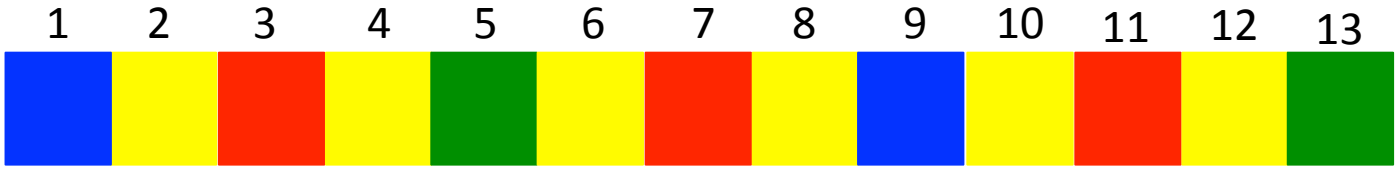


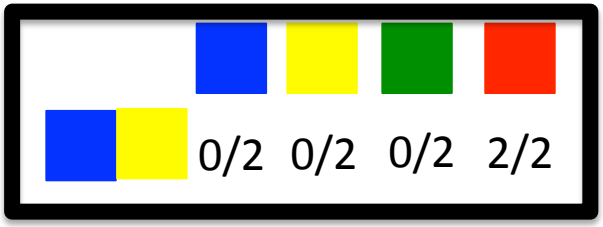


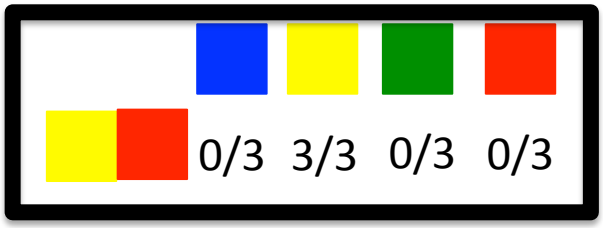


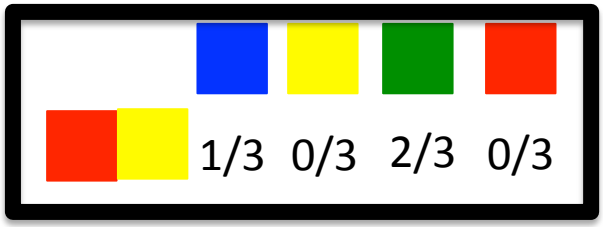
Completăm textura

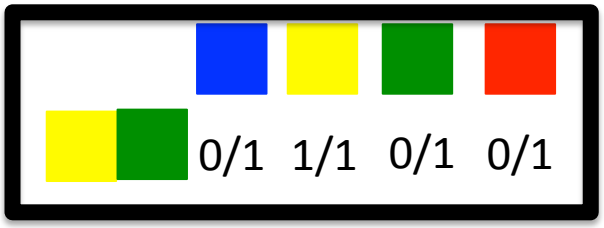
Completăm textura pixel cu pixel după următoarea regulă: observăm cei doi vecini din stânga (contează ordinea) si apoi selectăm un pixel din distribuția condiționată (fără să o updatăm).

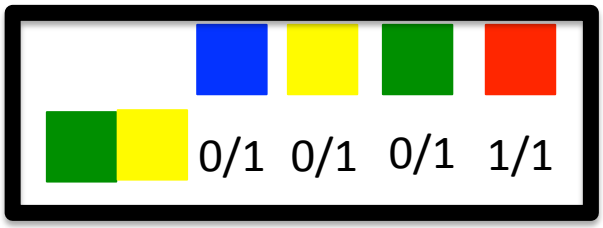


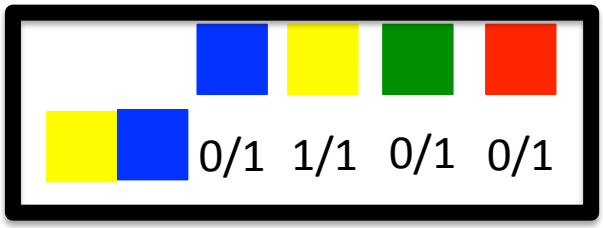












Lanțuri Markov

Lanț Markov:

- o secvență de variabile aleatoare **x_1, x_2, …, x_n**
- **x_t** reprezintă **starea** modelului la momentul t

- **Ipoteza de bază**: fiecare stare x_t depinde numai de starea precedentă x_{t-1}
  - dependența este dată de o **probabilitate condiționată:**

$$p(\mathbf{x}_t|\mathbf{x}_{t-1})$$

- Exemplul de mai sus: lanț Markov de ordinul 1
- Lanț Markov de ordinul N:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-N})$$

Slide adaptat după K. Grauman

Câmp Markov aleator

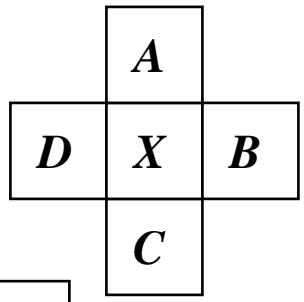
Câmp Markov aleator

- Generalizare a lanțurilor Markov în 2 sau mai multe dimensiuni

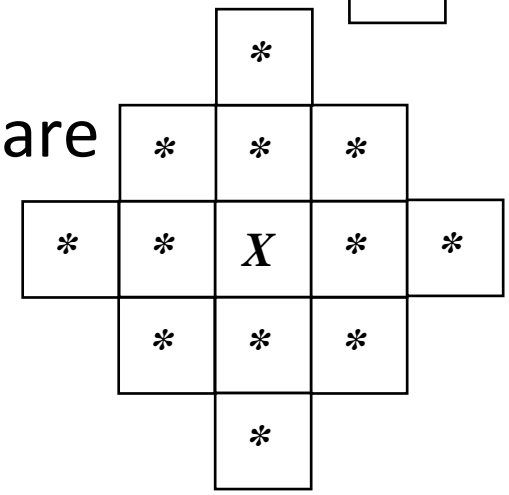
Câmpuri Markov aleatoare de ordinul 1:

- probabilitatea ca un pixel *X* ia o anumită valoare depinde de valorile pixelilor vecini *A*, *B*, *C*, si *D*:

$$P(\mathbf{X}|\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$$



Câmpuri Markov aleatoare de ordin superior



Slide adaptat după K. Grauman

Generarea texturii

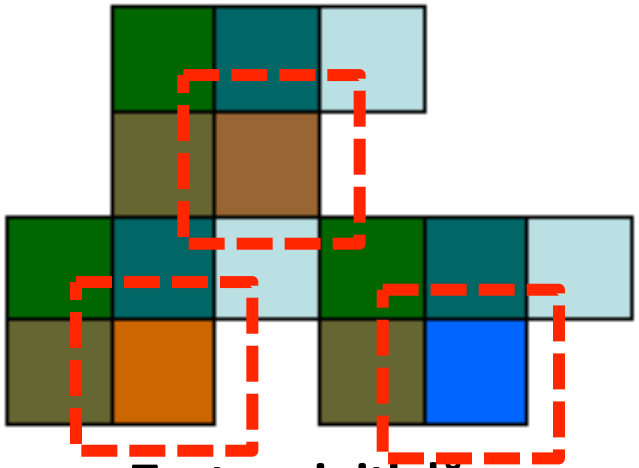
Textură inițială

Textură generată

Alexei A. Efros and Thomas K. Leung, “Texture Synthesis by Non-parametric Sampling,” ICCV, 1999.

Generarea texturii – intuiție

Inserăm pixeli cu o anumită intensitate pe baza valorilor pixelilor vecini



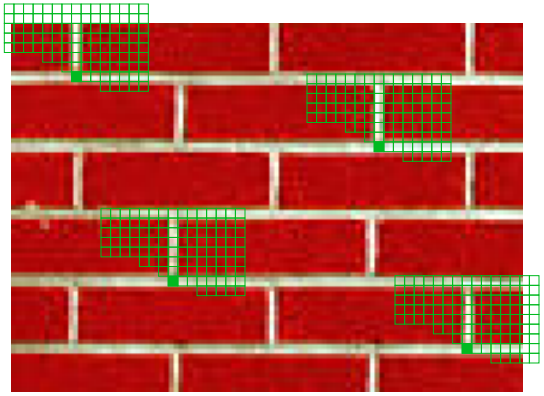
Textura inițială

Vrem să inserăm un nou pixel aici

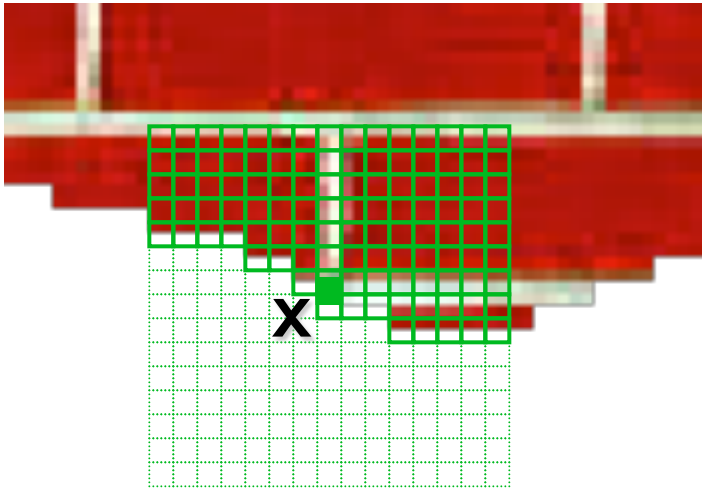
Distribuția intensității unui pixel este condiționată de vecinii lui (pentru acest exemplu).

Slide adaptat după A. Efros

Generarea unui singur pixel



Textura inițială

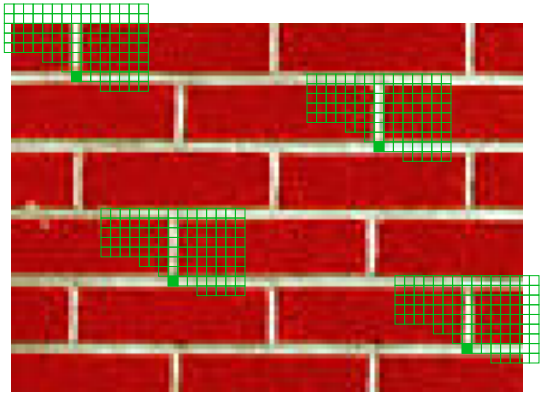


Imagine generată

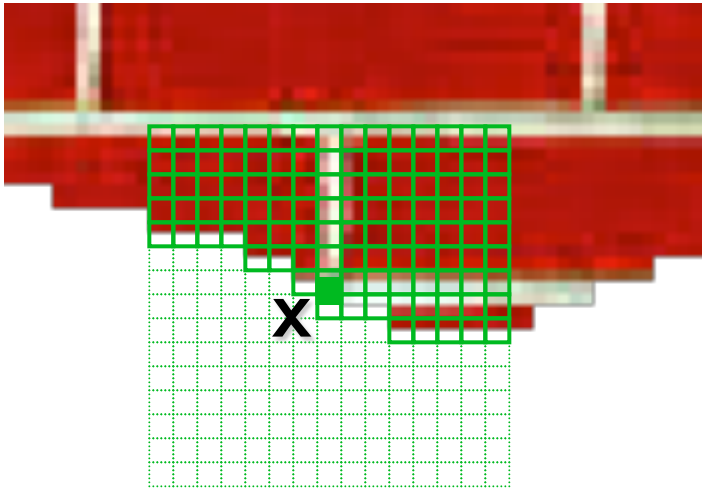
- Care este P(x|vecinătatea lui x)?
- Găseşte toate ferestrele din textura inițială care seamănă perfect cu vecinătatea lui x
- Pentru a genera **x**
  - alege aleator o fereastă din cele găsite
  - valoarea lui **x** = valoarea pixelului central al ferestrei

Slide adaptat după A. Efros

Generarea unui singur pixel



Textura inițială



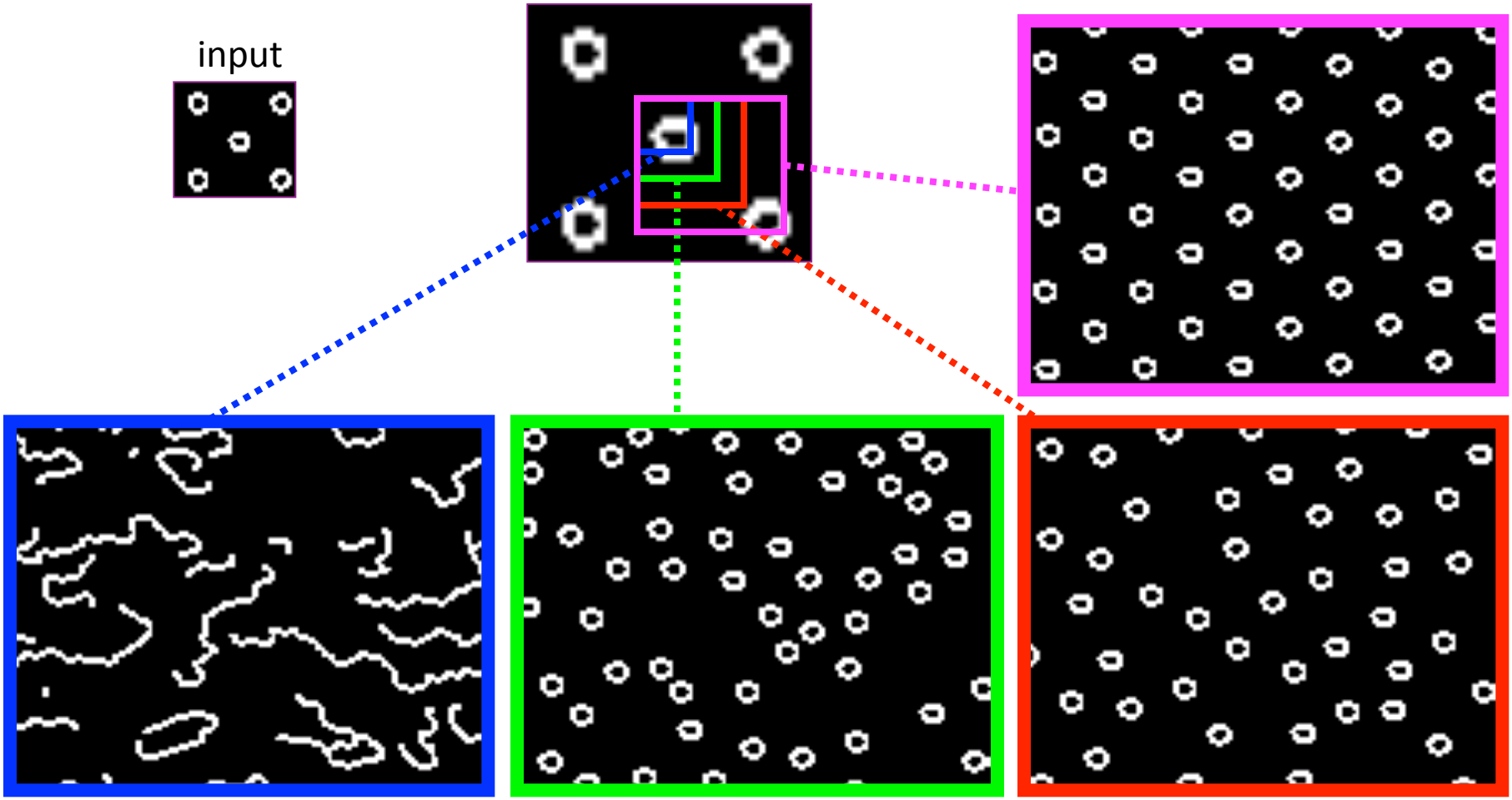
Imagine generată

- E posibil să nu existe nicio fereastră care seamănă perfect cu vecinătatea lui x.
- Găseşte cele mai apropiate ferestre pe baza unei distanțe (suma pătratelor distanțelor) si alege una din aceste ferestre pe baza distanței
- Transformă distanța în probabilitate astfel încât:

  distanțe mari = prob. mică, distanțe mici = prob. mare

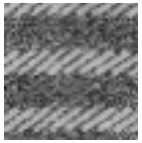
Slide adaptat după A. Efros

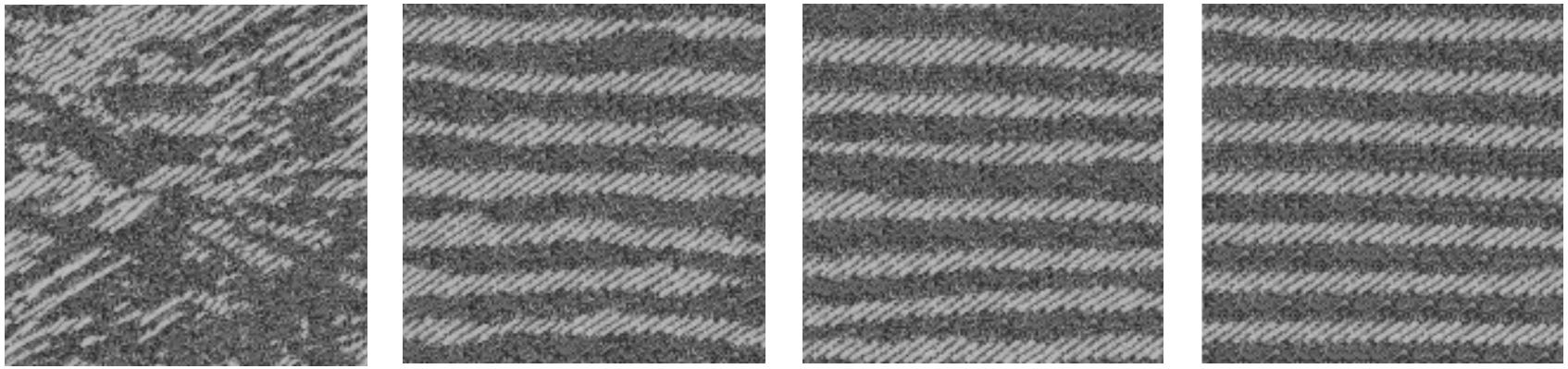
Dimensiunea ferestrei vecinătății

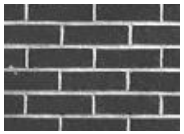


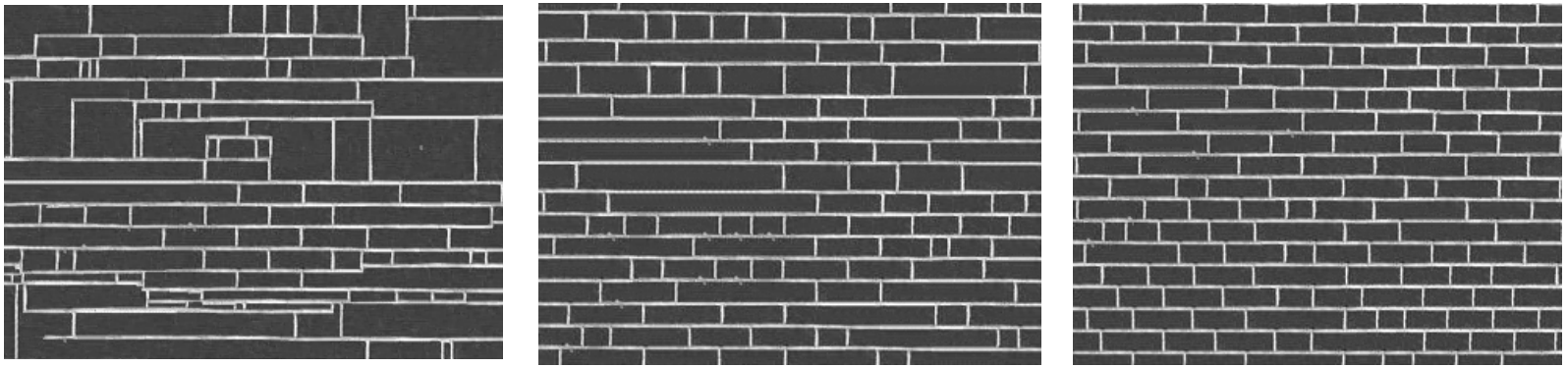
Slide adaptat după A. Efros

Dimensiunea ferestrei vecinătății





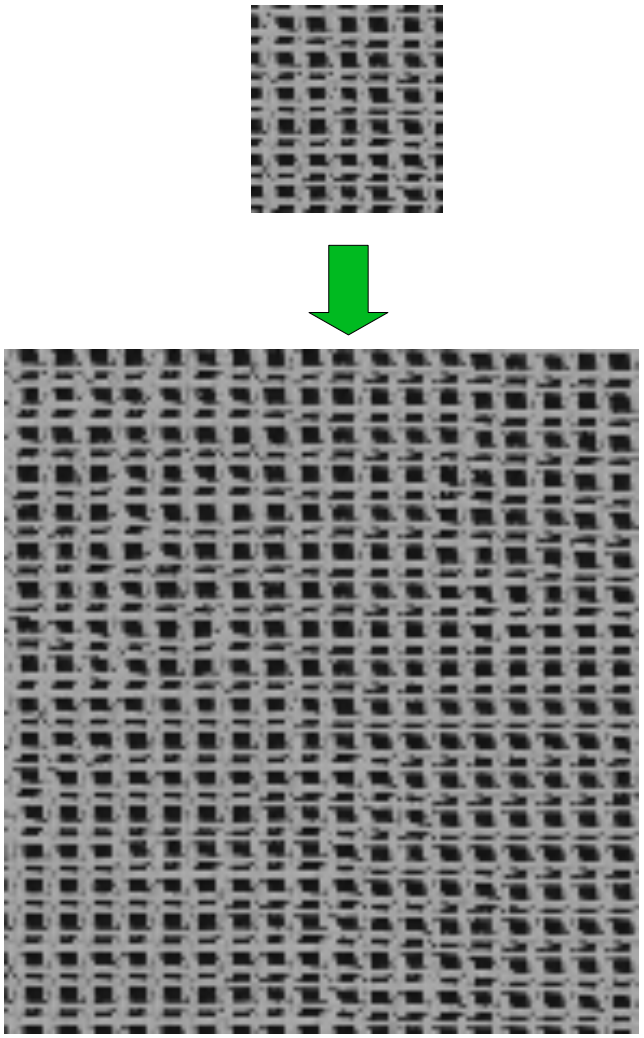


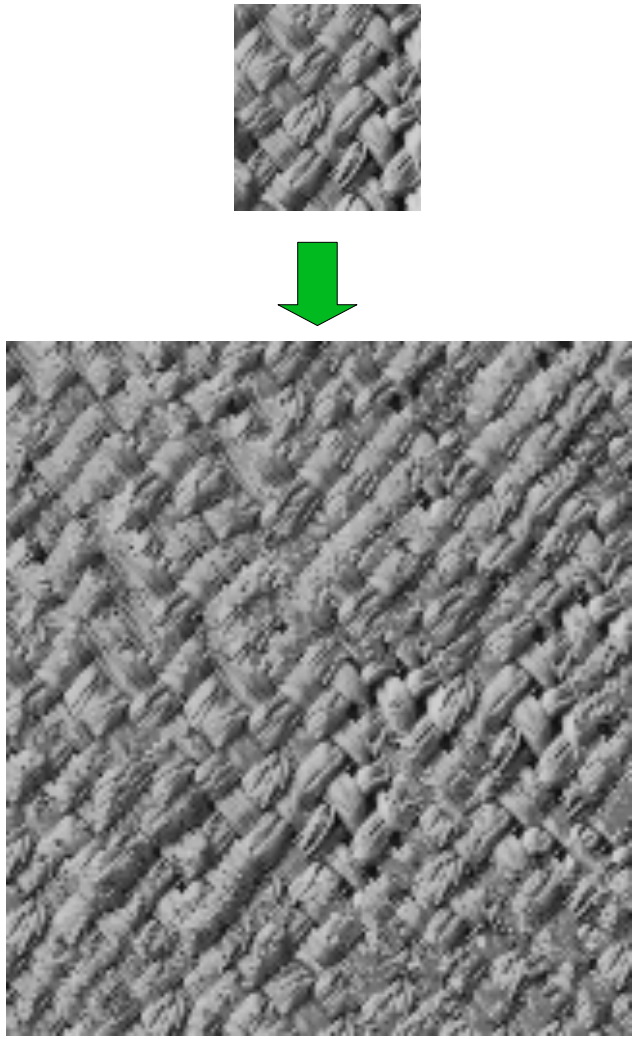


Creştem dimensiunea ferestrei

Slide adaptat după A. Efros

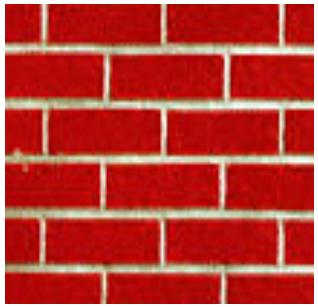
Generarea texturii - rezultate

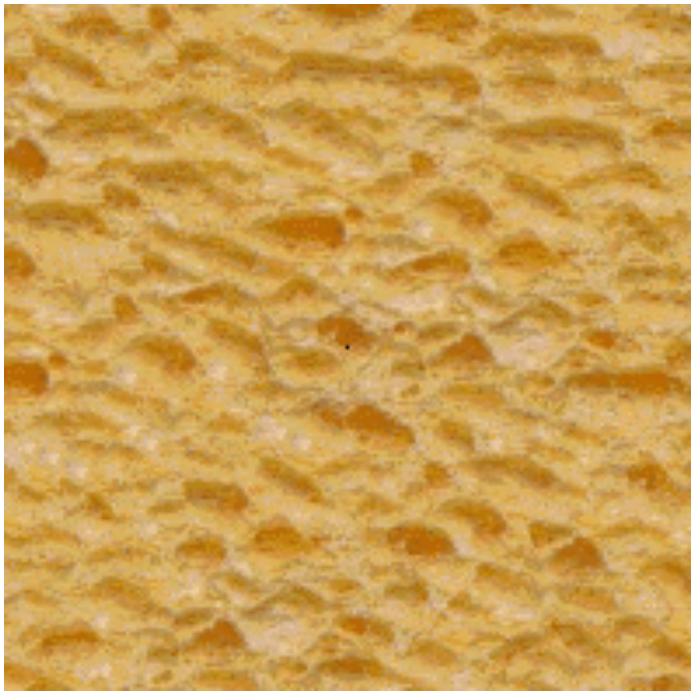


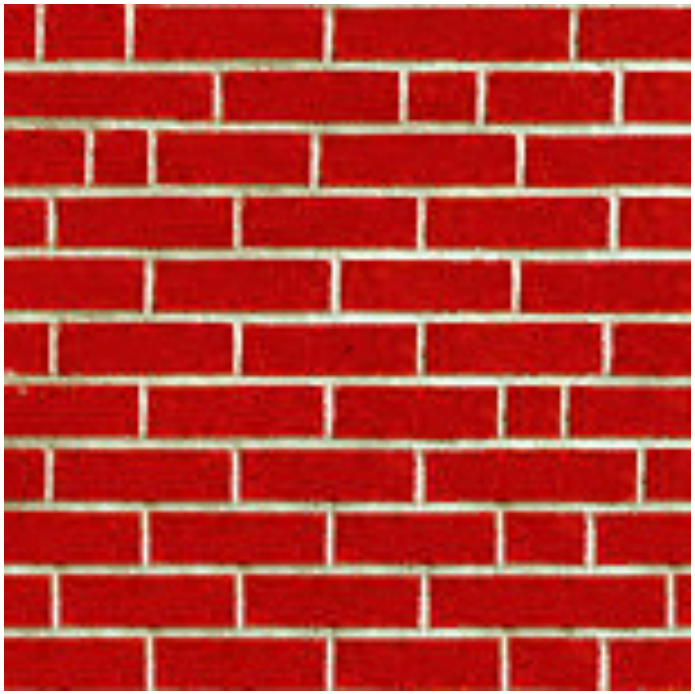


Slide adaptat după A. Efros

Generarea texturii - rezultate

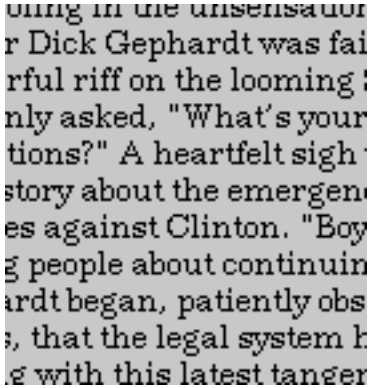


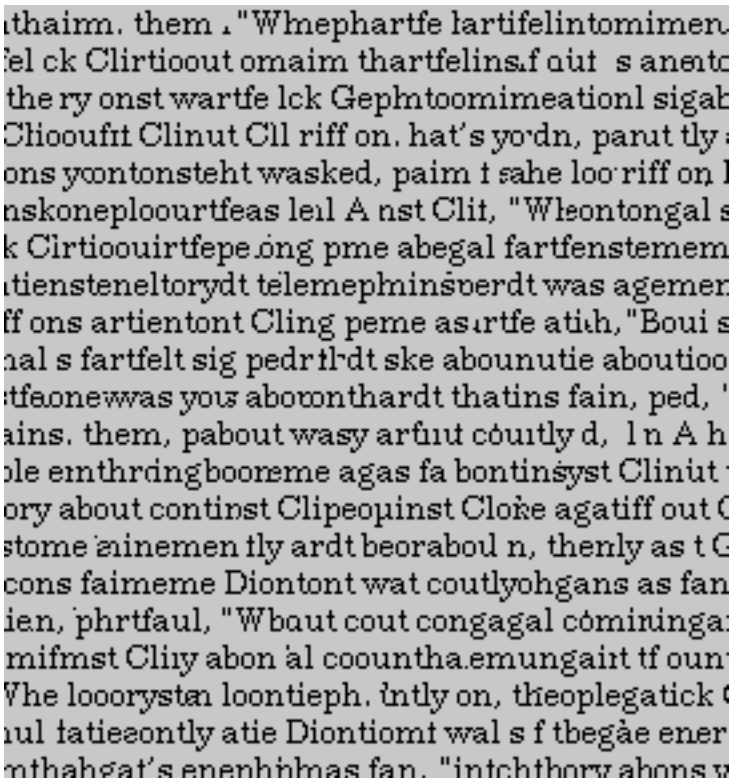




Slide adaptat după A. Efros

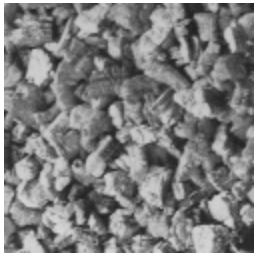
Generarea texturii - rezultate

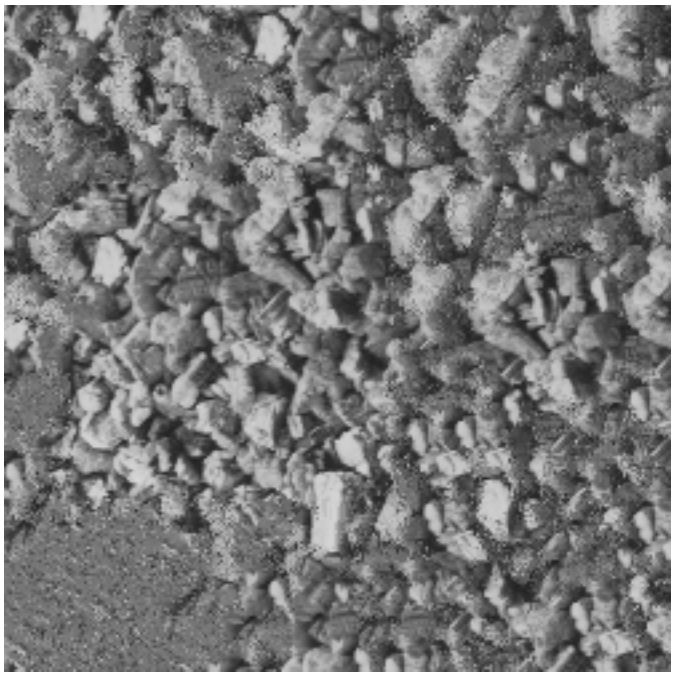




Slide adaptat după A. Efros

Cazuri nereuşite



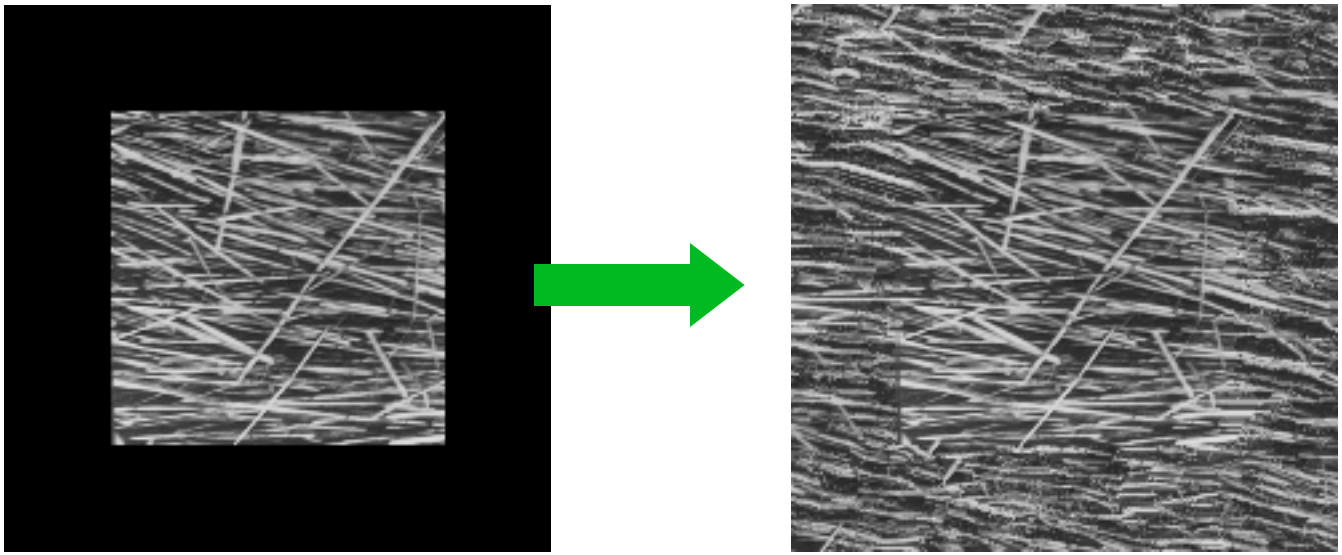


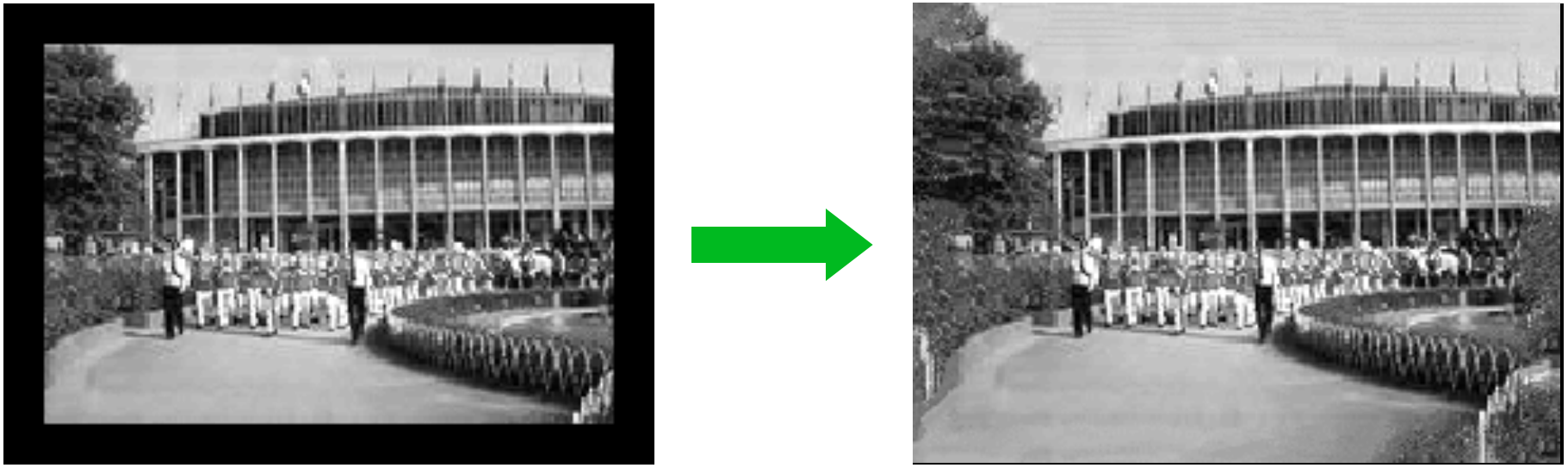
texteli care nu sunt
În textura inițială

Copiază

Slide adaptat după A. Efros

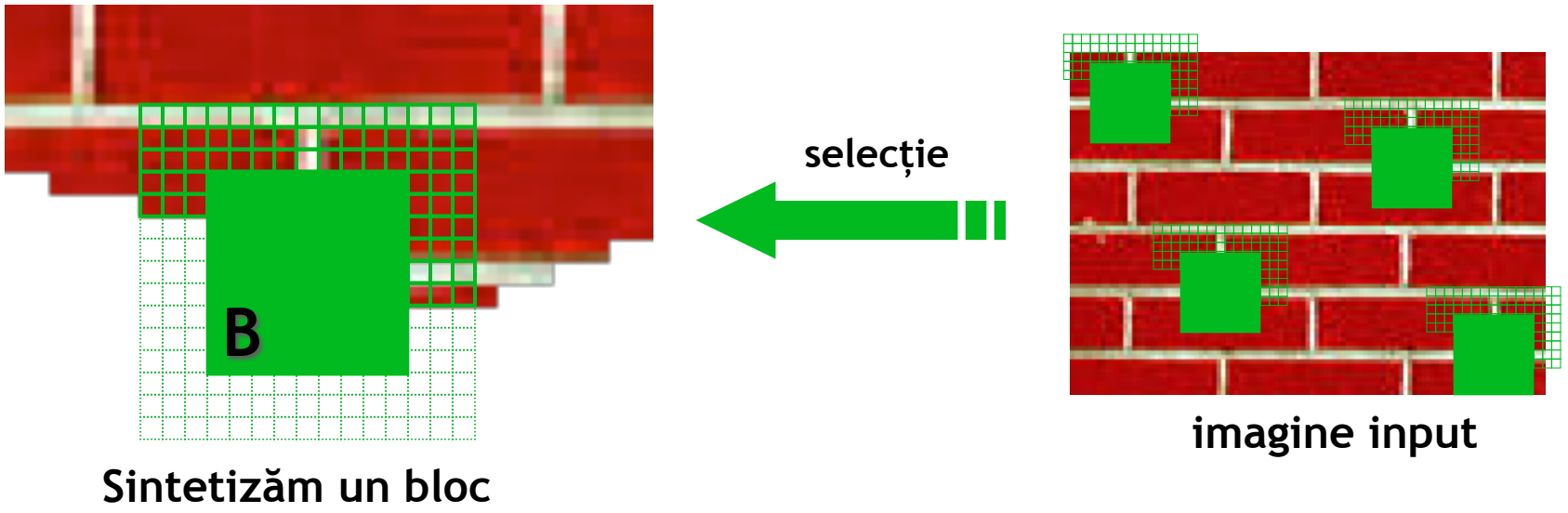
Extrapolare





Slide adaptat după A. Efros

Generarea texturii la nivel de blocuri

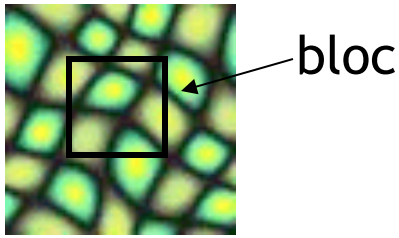


- Observație: pixelii vecini sunt foarte corelați

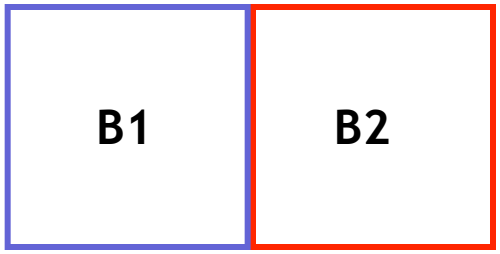
Idee: unitate de sinteză = bloc de pixeli

- Înlocuim un pixel cu un bloc : P(**B**|Vecinătate(**B**))
- Mult mai rapid: sintetizăm toți pixelii dintr-un bloc odată

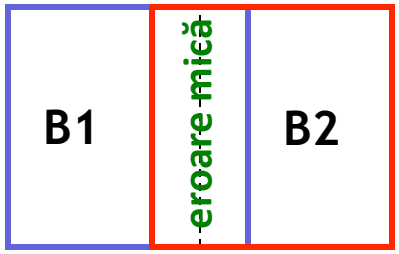
Slide adaptat după A. Efros



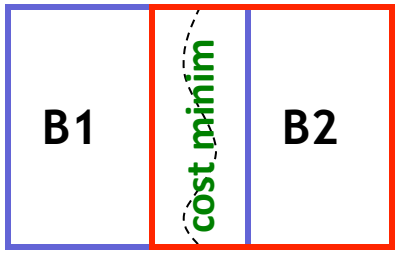
textură



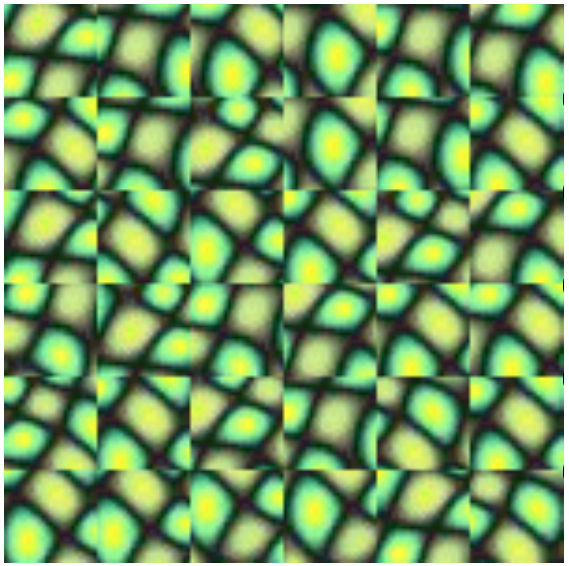
blocuri aleatoare din textură
puse unele lângă altele

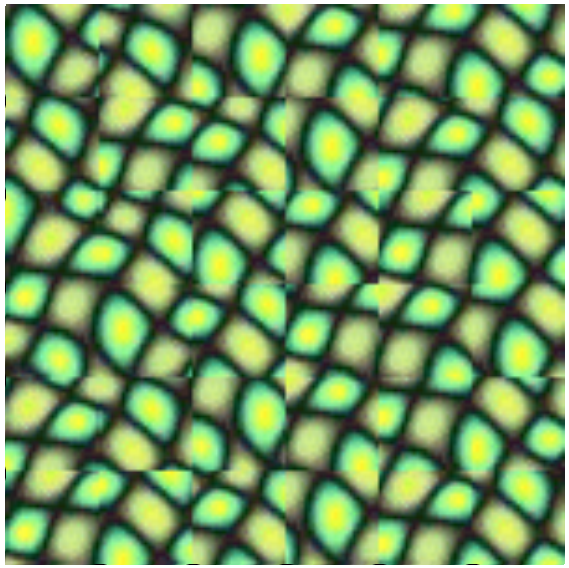


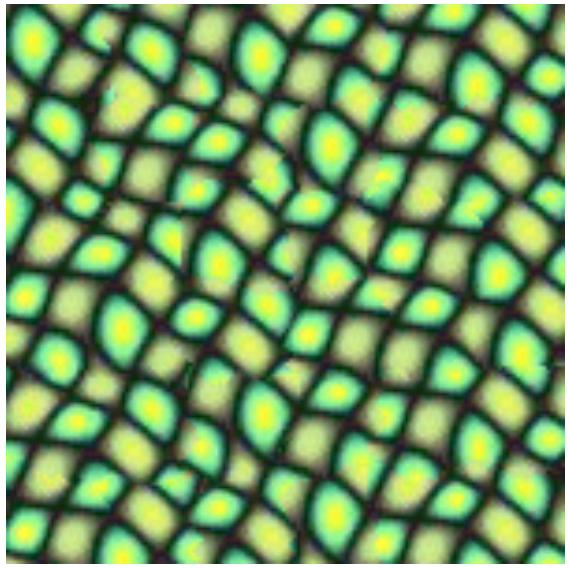
blocuri aleatoare din textură
care se suprapun la frontieră



blocuri aleatoare din textură
care se suprapun la frontieră
frontiera de cost minim



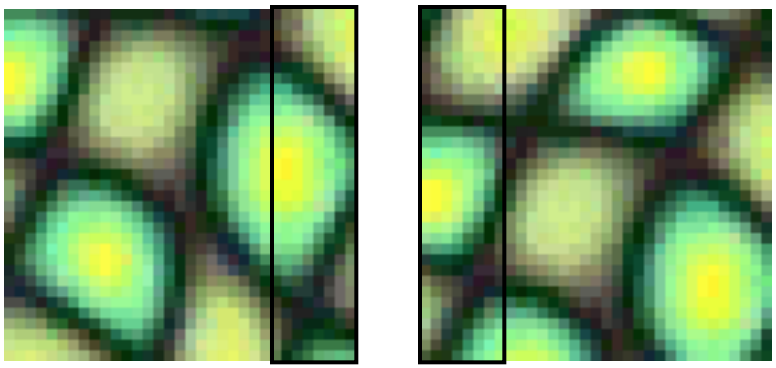




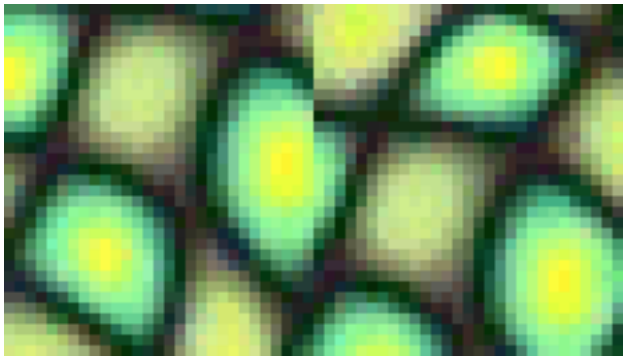
Slide adaptat după A. Efros

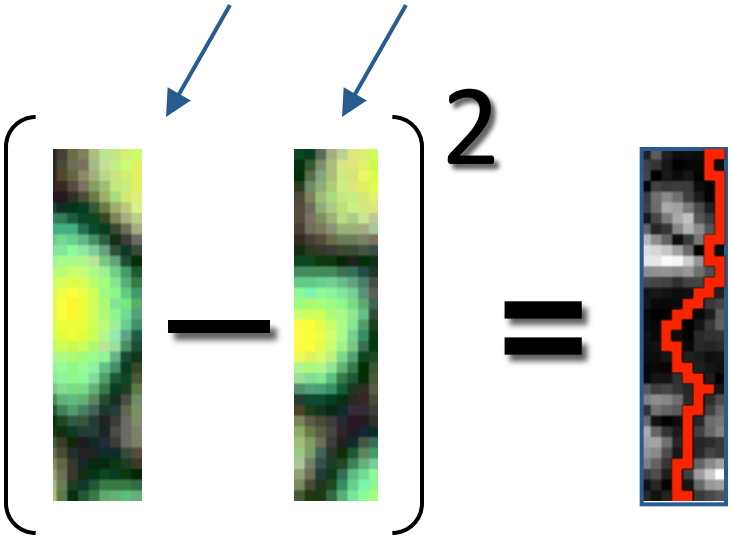
Frontiera de eroare minimă

blocuri care se suprapun

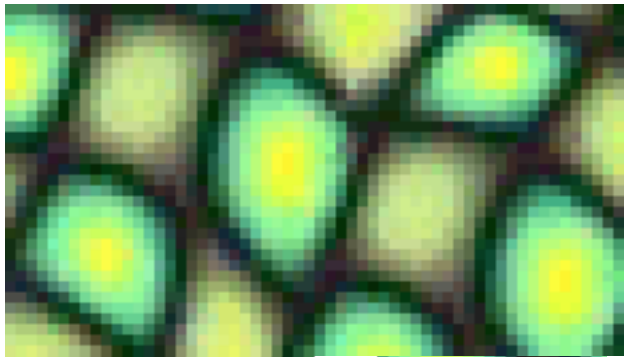


muchie/frontieră verticală





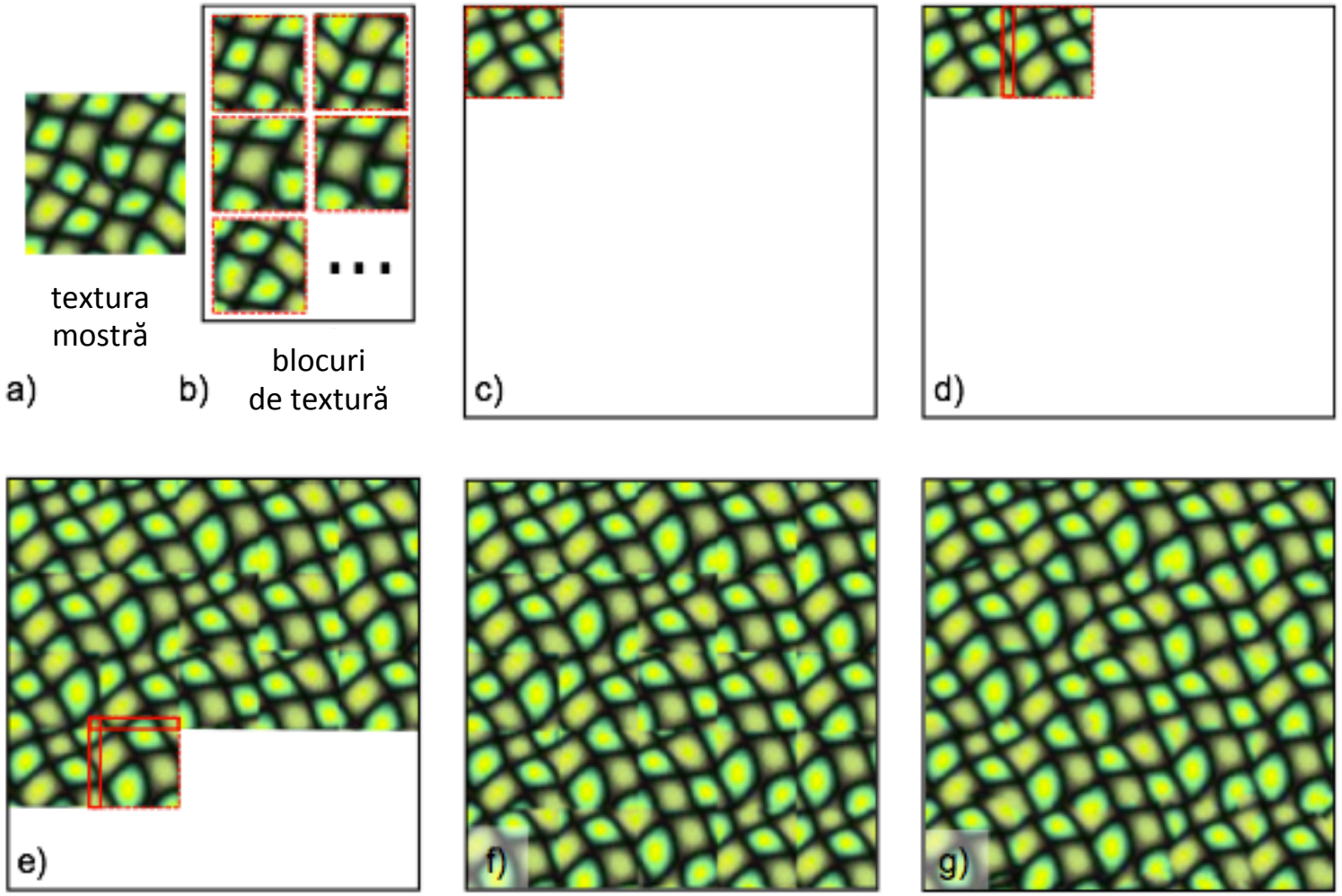
eroarea de suprapunere



frontiera de eroare minimă

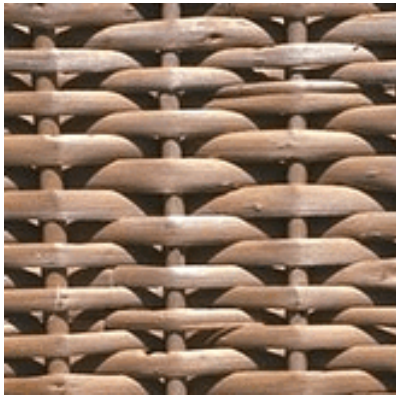
Slide adaptat după A. Efros

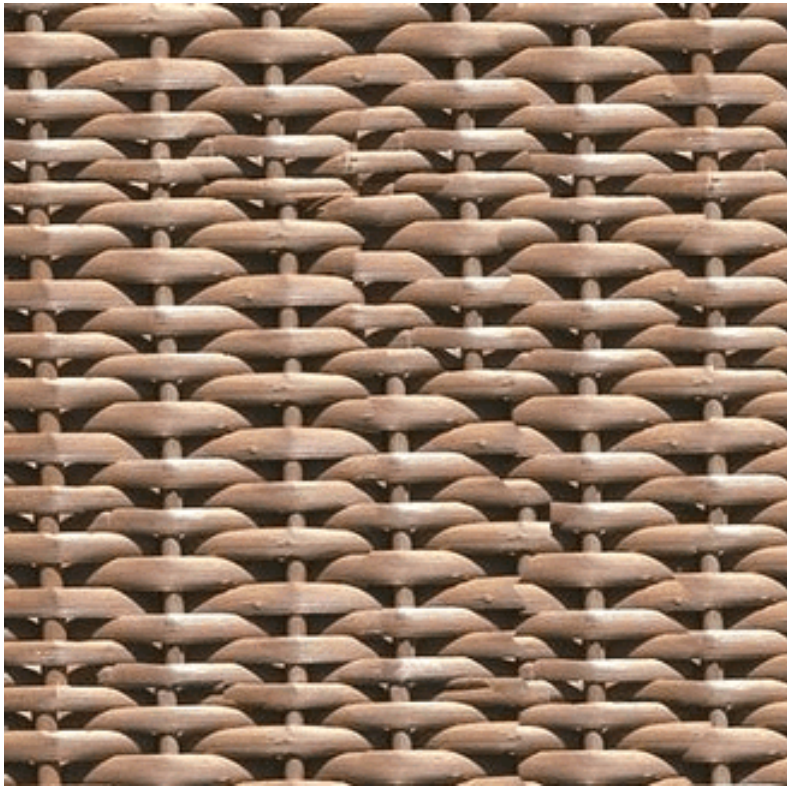
Sinteza texturii la nivel de blocuri - algoritm



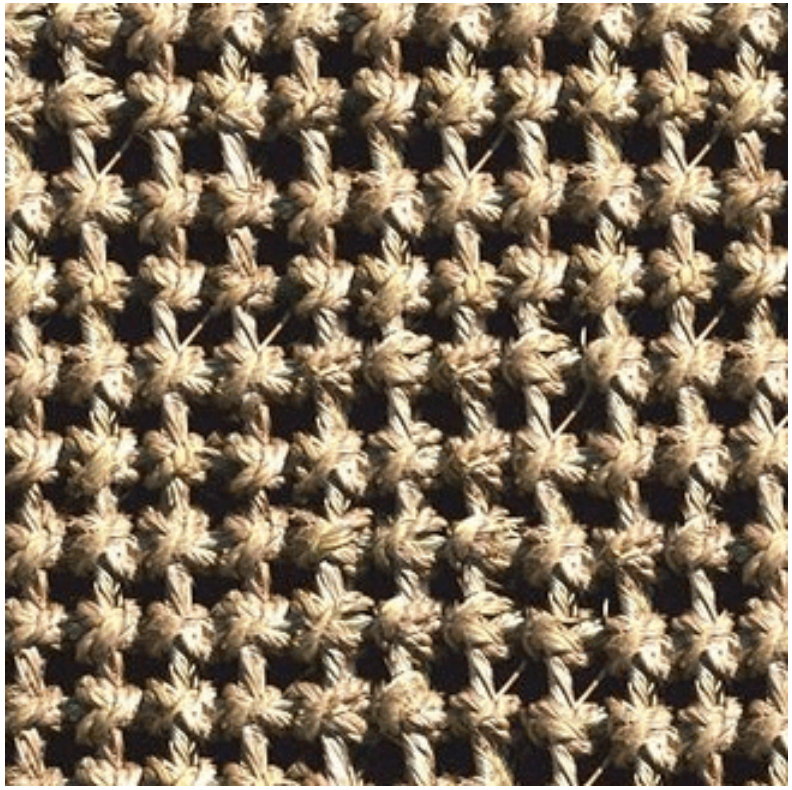
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/papers/efros-siggraph01.pdf

Slide adaptat după A. Efros





Slide adaptat după A. Efros

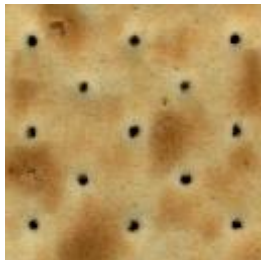
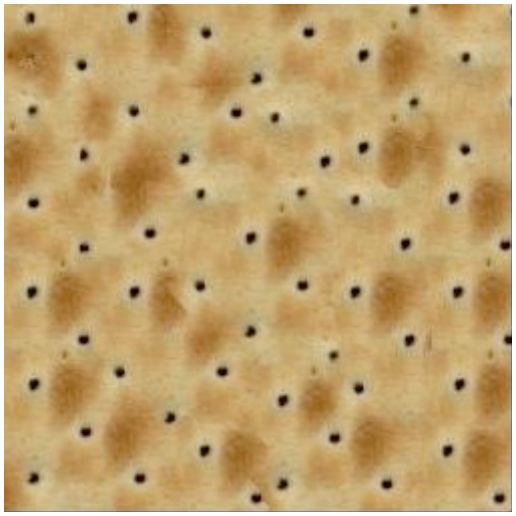


Slide adaptat după A. Efros

Slide adaptat după A. Efros

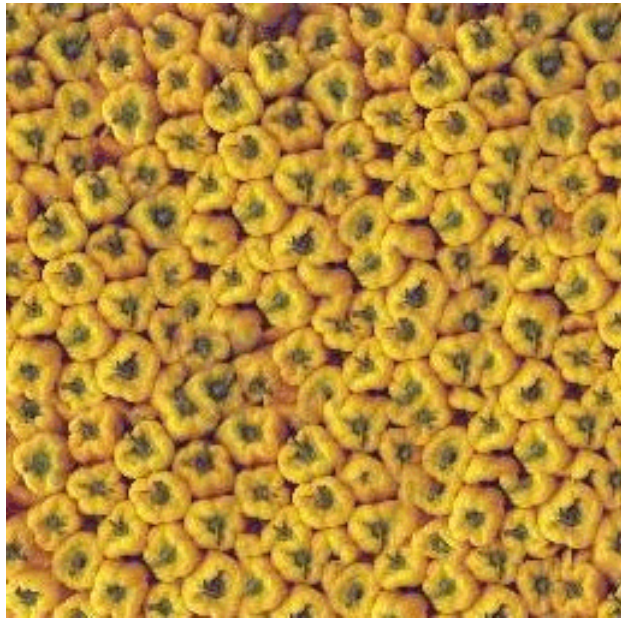
Slide adaptat după A. Efros

Slide adaptat după A. Efros





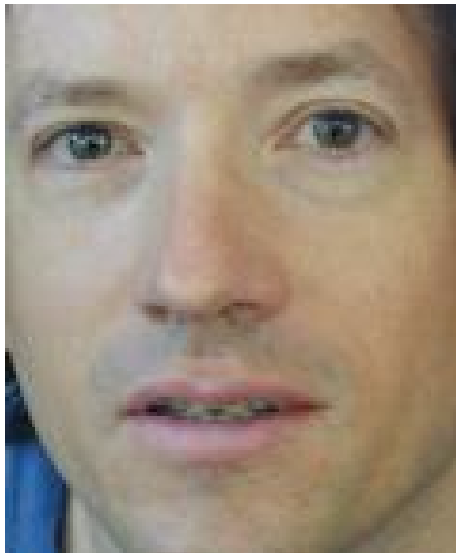


Slide adaptat după A. Efros

Mai puțin reușite

Slide adaptat după A. Efros

Transferul texturii



+

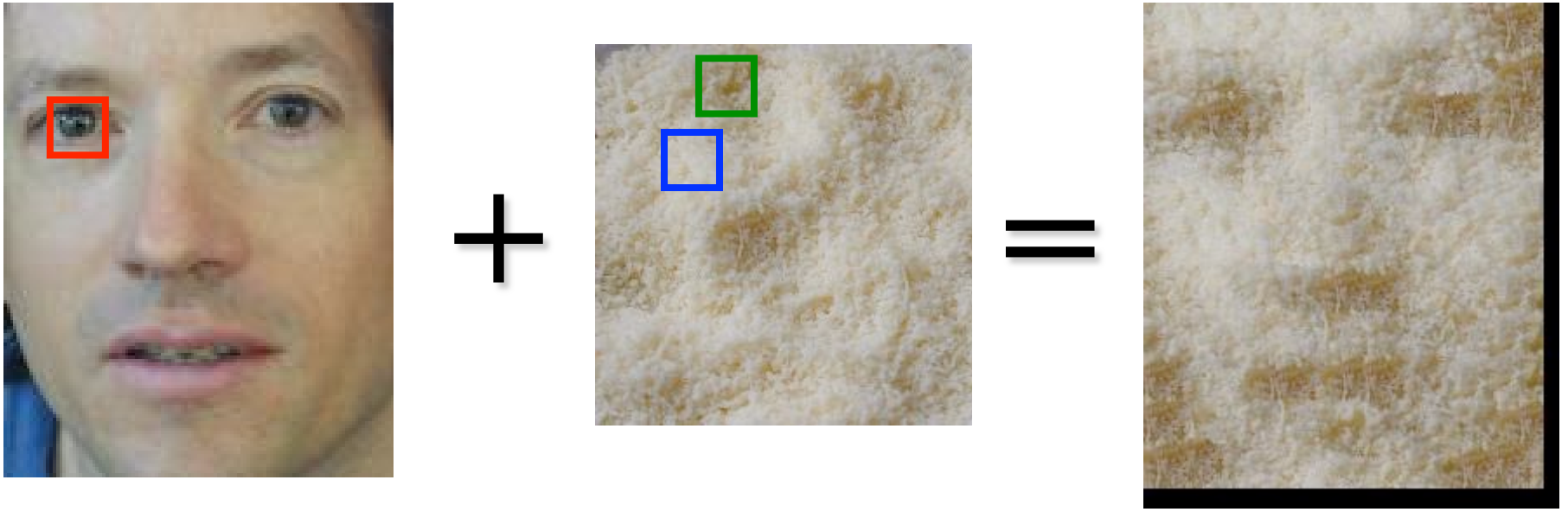
orez

=

Slide adaptat după A. Efros

Transferul texturii

- Transferăm textura unui obiect la un alt obiect



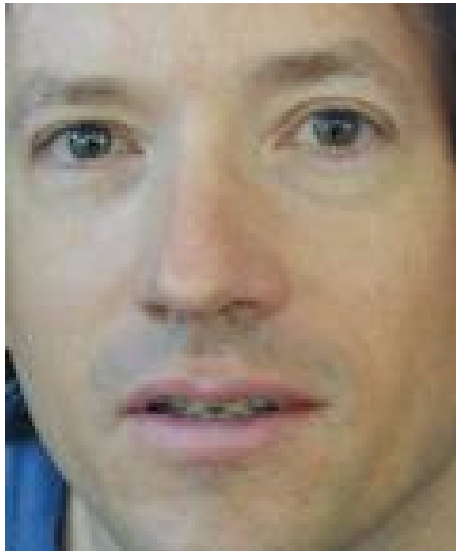
- Adăugăm un nou termen: similaritatea dintre blocul de textură si blocul din imagine
- Spre exemplu similaritate în termeni de intensitate

Slide adaptat după A. Efros

parmezan

+

=

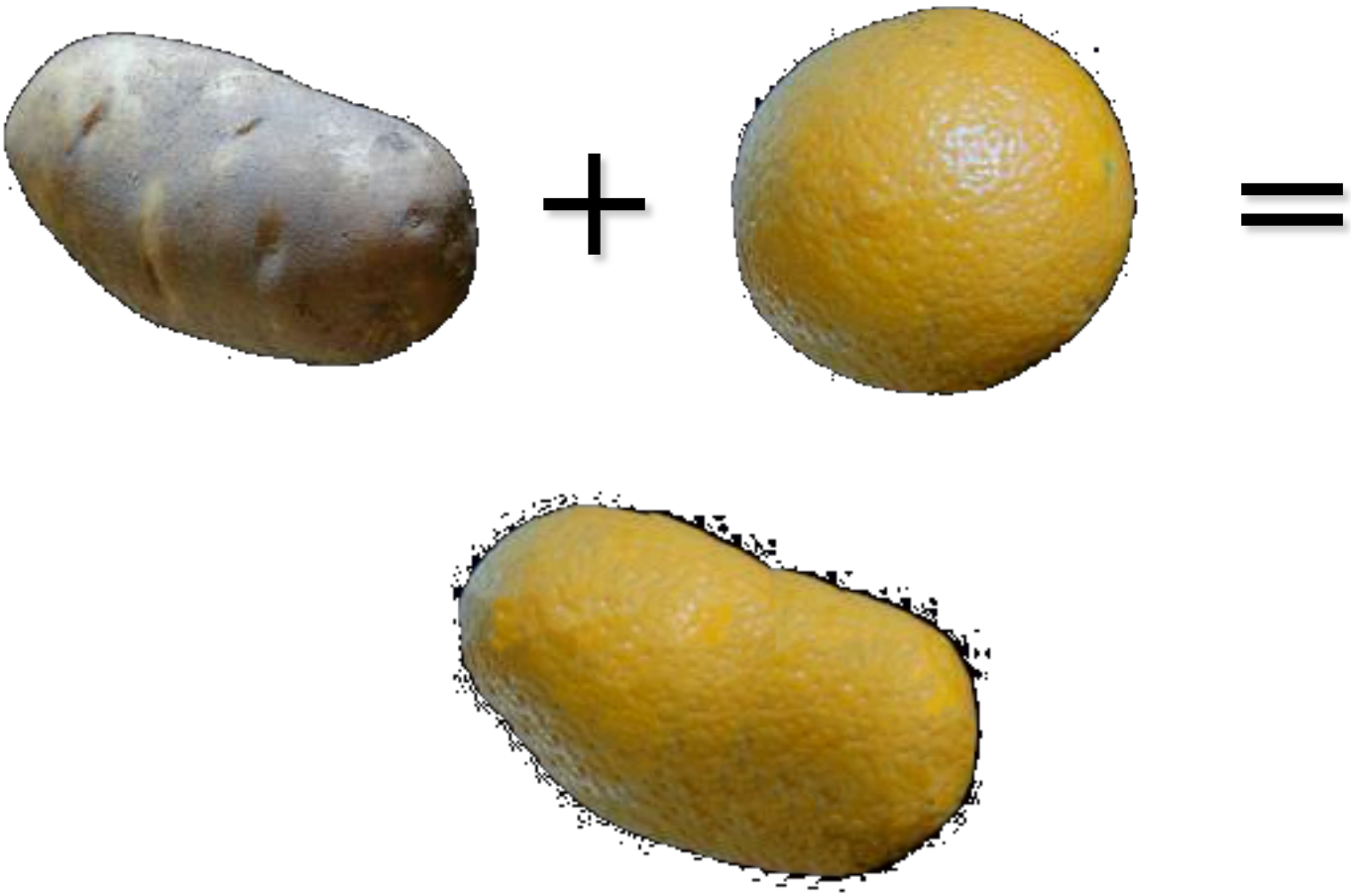


orez

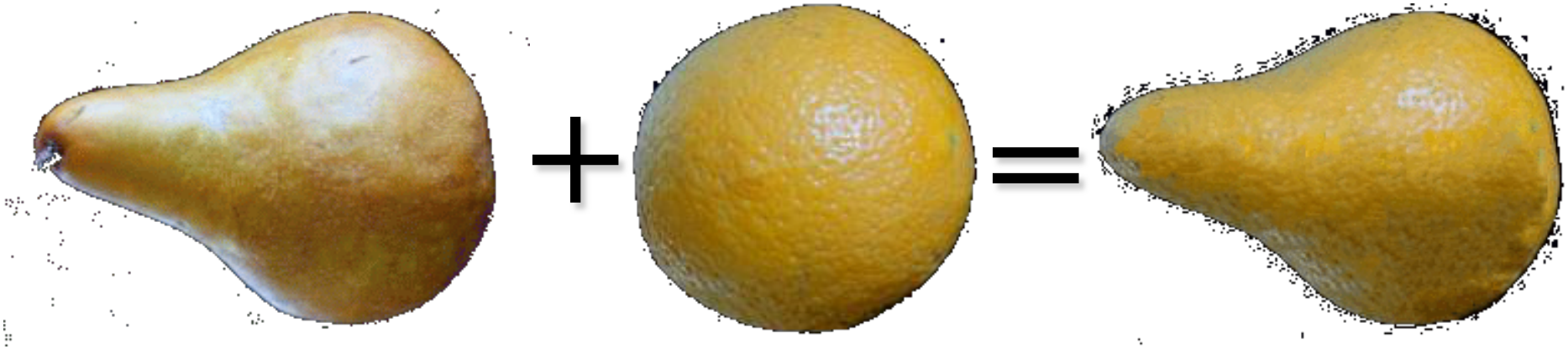
+

=

Slide adaptat după A. Efros



Slide adaptat după A. Efros



Slide adaptat după A. Efros