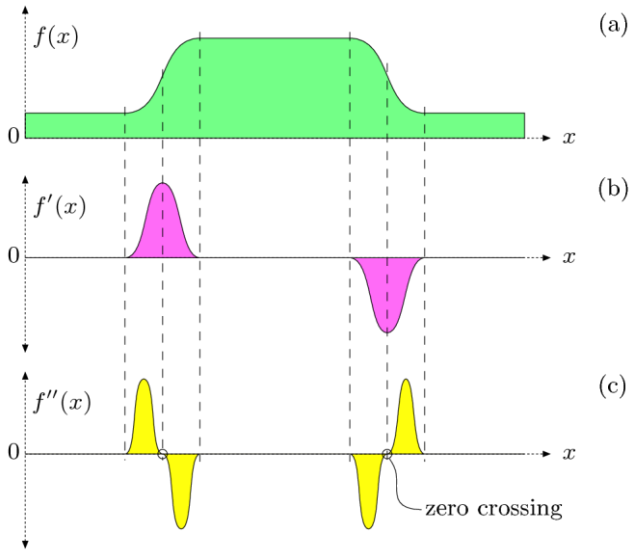
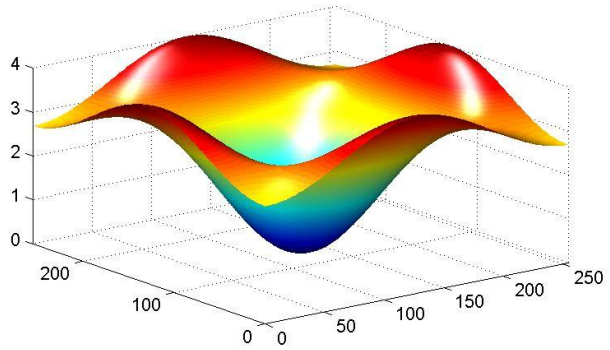
Laplaceov operator

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





$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -\mathbf{2} & 1 \end{bmatrix}$$

$$\frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -\mathbf{2} \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

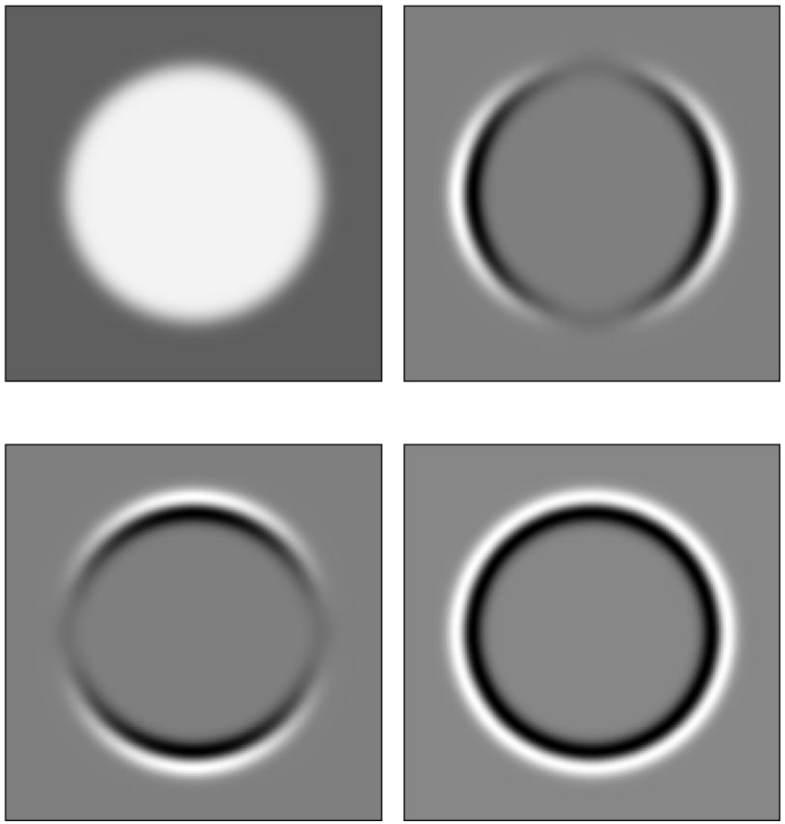
Laplaceov operator

- Filter ni separabilen, laho pa izkoristimo linearnost konvolucije

$$I * H^L = I * (H_x^L + H_y^L) = (I * H_x^L) + (I * H_y^L)$$

Laplaceov operator

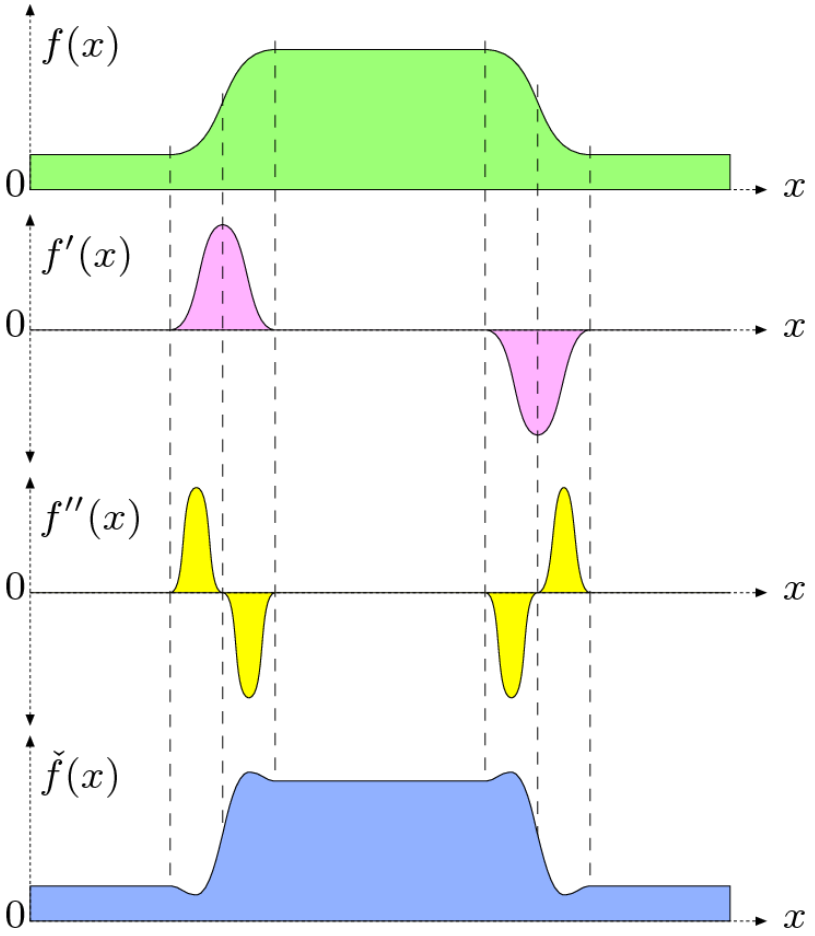
$$I * H^L = I * (H_x^L + H_y^L) = (I * H_x^L) + (I * H_y^L)$$



Ostrenje (sharpening) z Laplaceovim operatorjem

- Sliki odštejemo vrednost drugega odvoda slike

$$\check{f}(x) = f(x) - w \cdot f''(x)$$



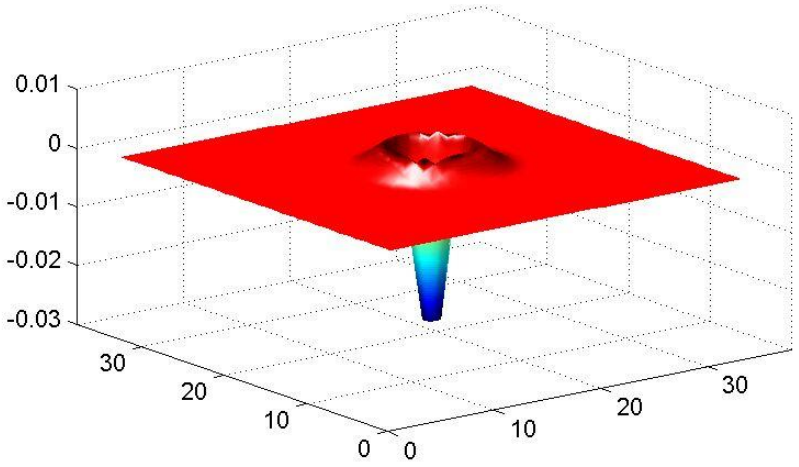
Laplace Gaussa (LoG)

$$(I * G) * H^L = I * (G * H^L)$$

$$\mathrm{LoG}_\sigma(x, y) = -\left(\frac{x^2 + y^2 - \sigma^2}{\sigma^4}\right) \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

on / off
center / surround
“mexican hat” valček

karakteristika LoG podobna karakteristiki nekaterih nevronov v vidnem korteksu



Razlika Gaussov (DoG)

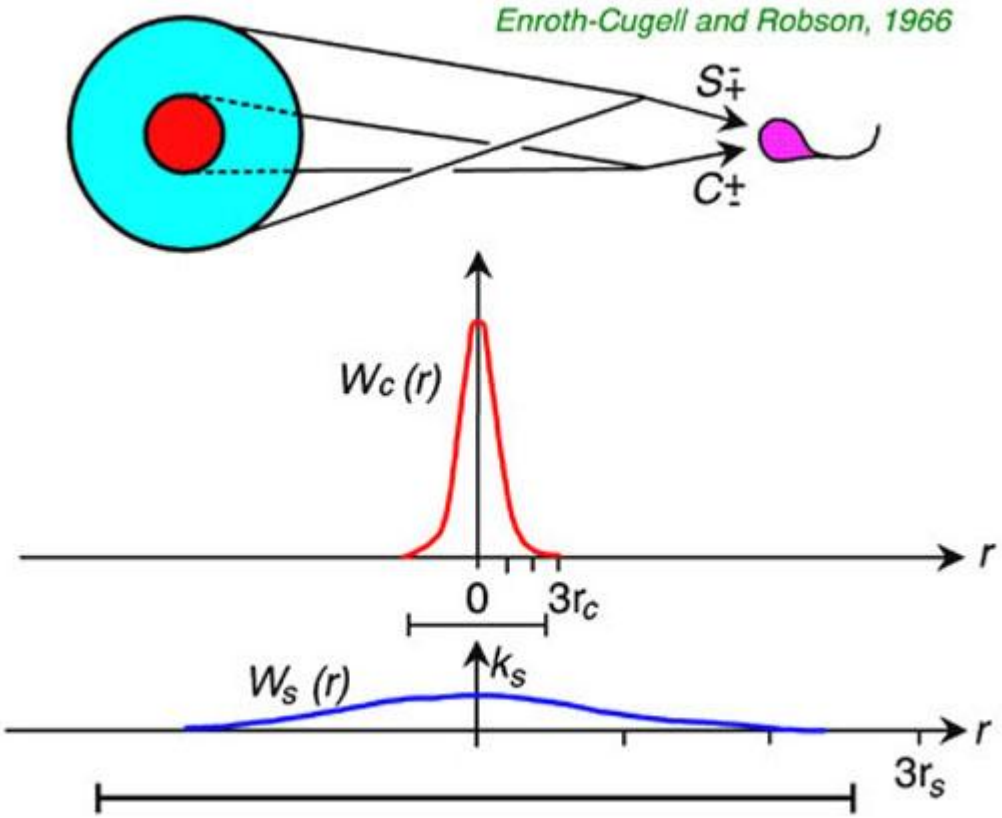


Fig. 10. Difference-of-Gaussians Receptive-Field Model (Enroth-Cugell and Robson, 1966).

Samodejna izbira merila v prostoru meril (Automatic scale selection)

- Vrednosti parcialnih odvodov slike v prostoru meril

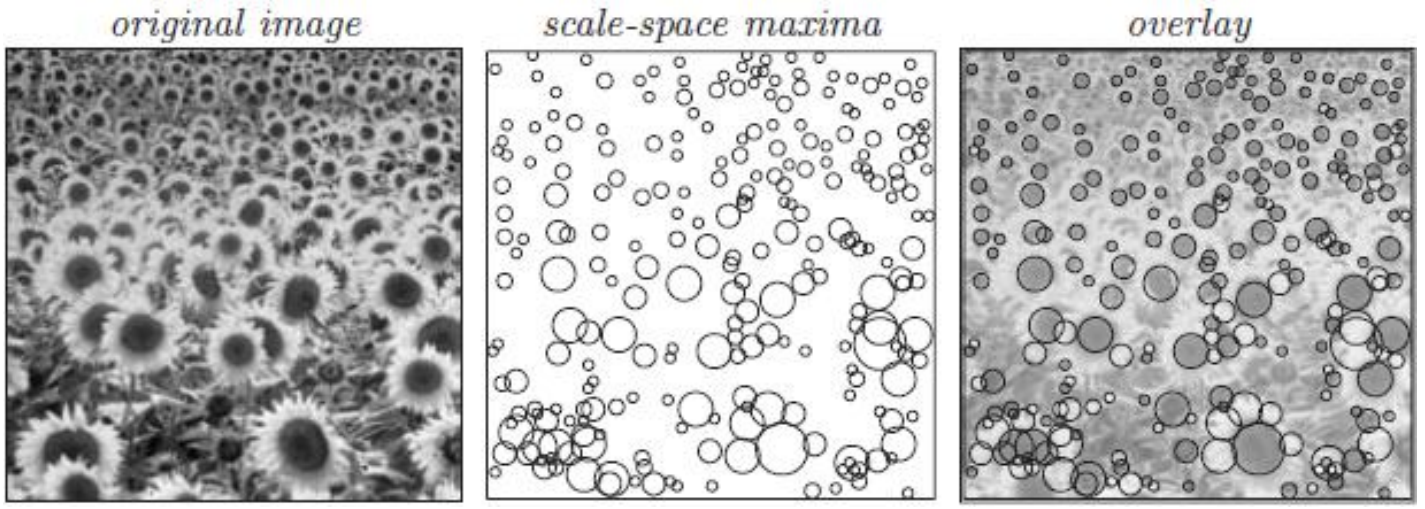
$$L(x, y, s\sigma); \quad s = 1, 2, \ldots, k$$

se z večanjem **s** manjšajo.

A general principle for automatic scale selection (T. Lindeberg)
In the absence of other evidence, assume that a scale level, at which some (possibly nonlinear) combination of normalized derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data.

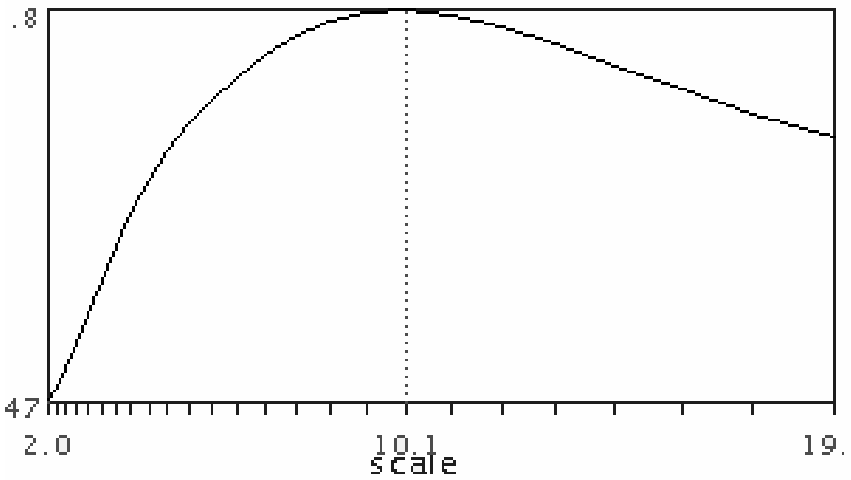
Samodejna izbira merila v prostoru meril

- Primer: maksimumi normaliziranega LoG operatorja $|s^2(H_x^L + H_y^L)|$ v prostoru meril



Samodejna izbira merila v prostoru meril

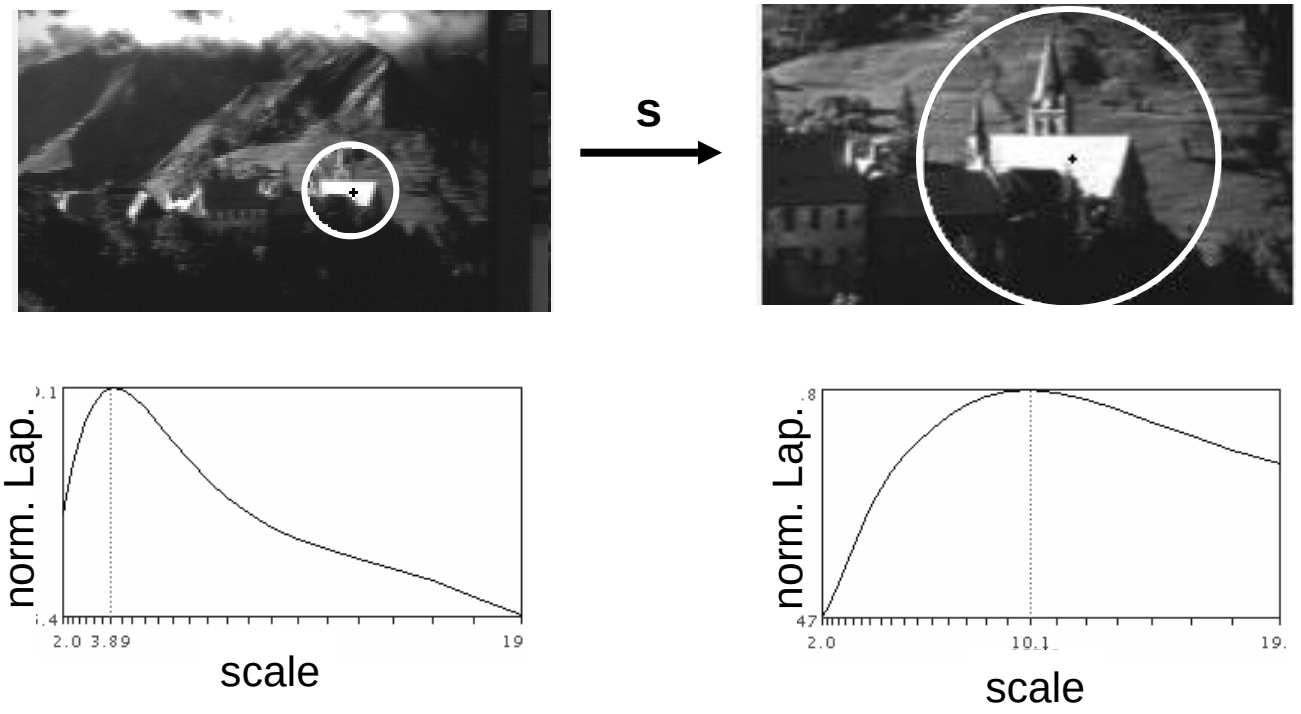
- Primer: maksimumi normaliziranega LoG operatorja $|s^2(H_x^L + H_y^L)|$ v prostoru meril



Samodejna izbira merila v prostoru meril

- značilno merilo

$$s \cdot s_1^* = s_2^*$$



Vir: Cordelia Schmid

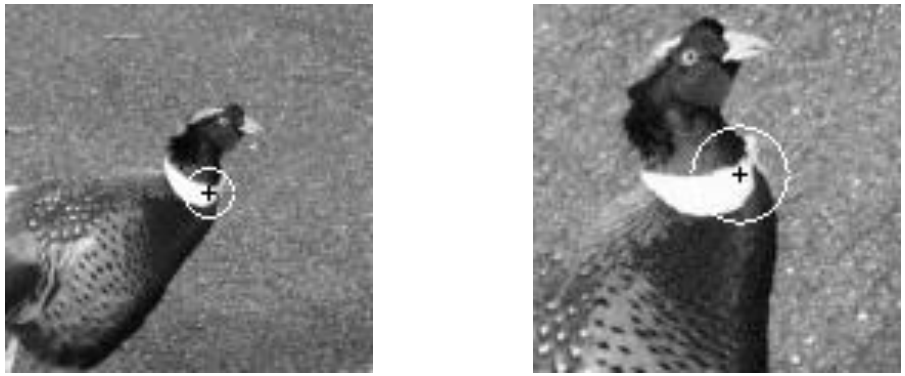
Harris - Laplaceov detektor oglišč

- Detektor regij, invarianten na spremembo merila

iz M_s pridobi oglišča C_s na $L_s(x, y, \sigma)$
za vsak $c \in C_s$
  if (c lokalni maksimum Lapl. operatorja v prostoru meril)
    c uvrsti na seznam oglišč
    merilo(c) = s

Harris - Laplaceov detektor oglišč

- Detektor regij, invarianten na spremembo merila



Vir: Cordelia Schmid

Primerjava lokalnih regij

SSD : $\frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}(I_1(x_1+i,y_1+j)-I_2(x_2+i,y_2+j))^2$

Robustnost na spremembe v osvetlitvi?

Globalno povečanje intenzitete: (I $\rightarrow$ I + b)

=> Normalizacija z povprečno vrednostjo regije

$$\frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}((I_1(x_1+i,y_1+j)-m_1)-(I_2(x_2+i,y_2+j)-m_2))^2$$

Sremembe kontrasta: (I $\rightarrow$ aI + b)

=> Normalizacija z povprečno vrednostjo regije in standardno deviacijo regije

$$\frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}\left(\frac{I_1(x_1+i,y_1+j)-m_1}{\sigma_1}-\frac{I_2(x_2+i,y_2+j)-m_2}{\sigma_2}\right)^2$$

Primerjava regij s križno korelacijo

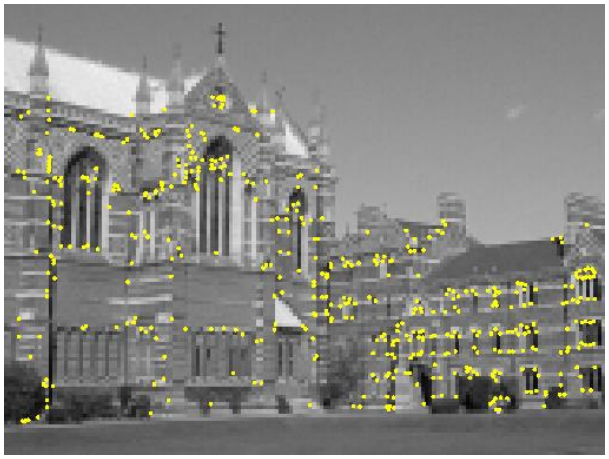
$$\frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}\left(\frac{I_1(x_1+i,y_1+j)-m_1}{\sigma_1}-\frac{I_2(x_2+i,y_2+j)-m_2}{\sigma_2}\right)^2$$

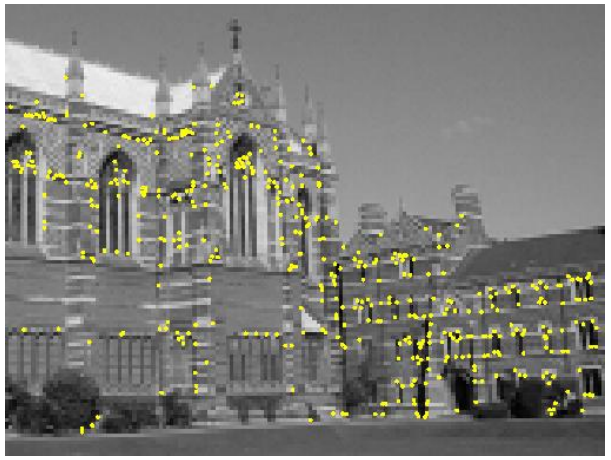
ZNCC:

$$\frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}\left(\frac{I_1(x_1+i,y_1+j)-m_1}{\sigma_1}\right)\cdot\left(\frac{I_2(x_2+i,y_2+j)-m_2}{\sigma_2}\right)$$

vrednosti med -1 in 1

Primerjava regij s križno korelacijo

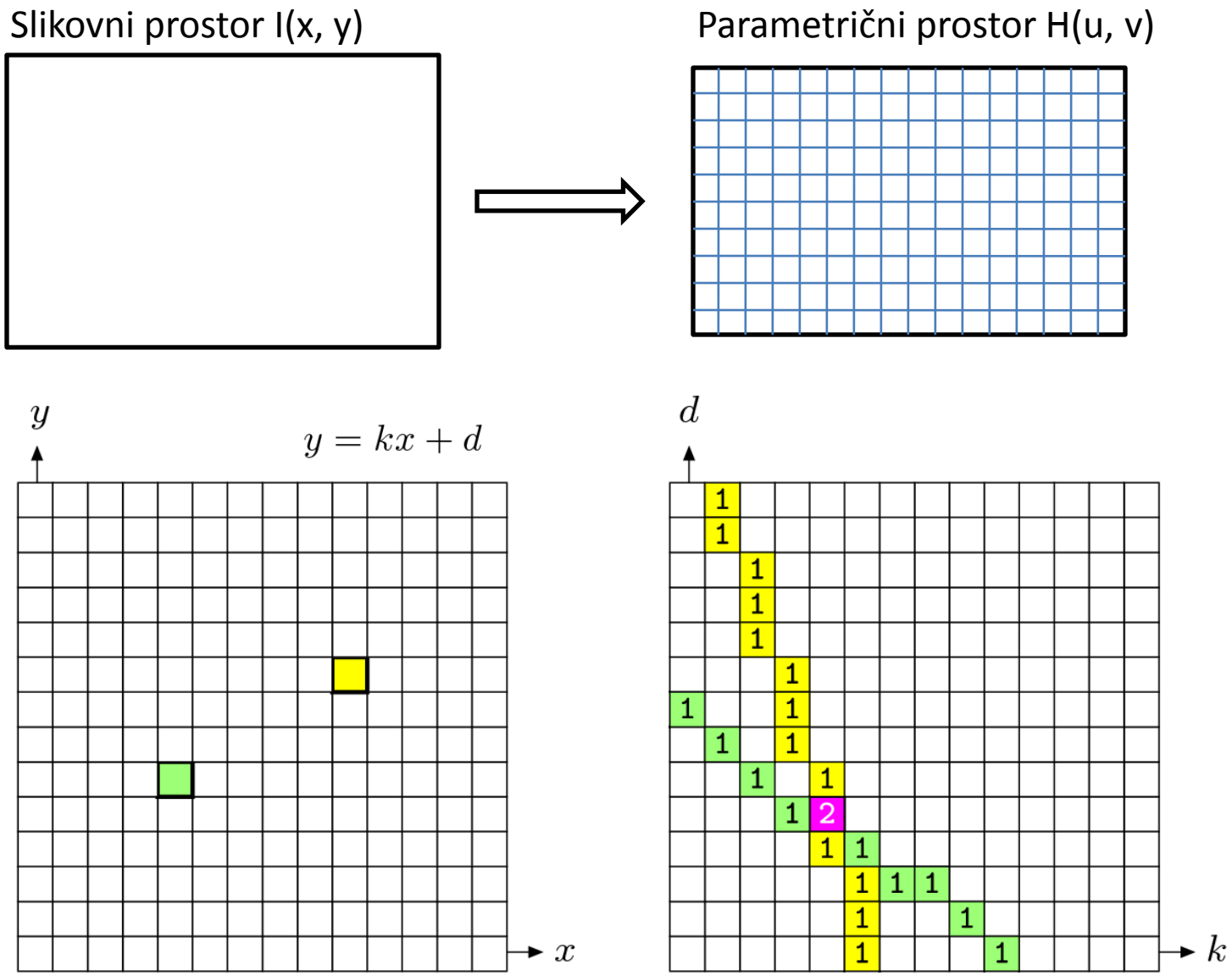




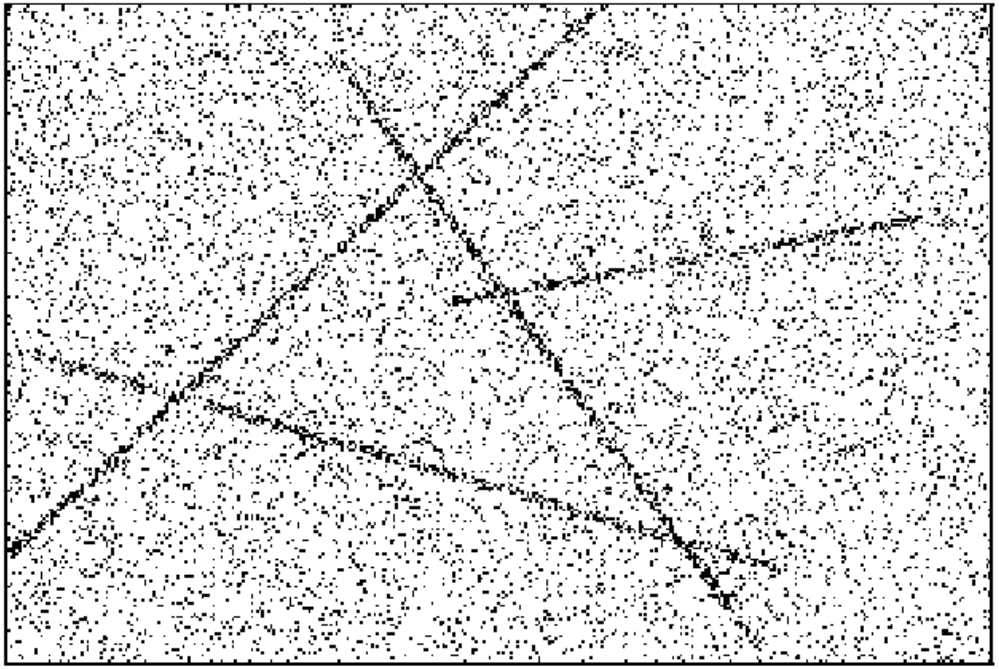
Vir: Cordelia Schmid

HOUGHOVA TRANSFORMACIJA

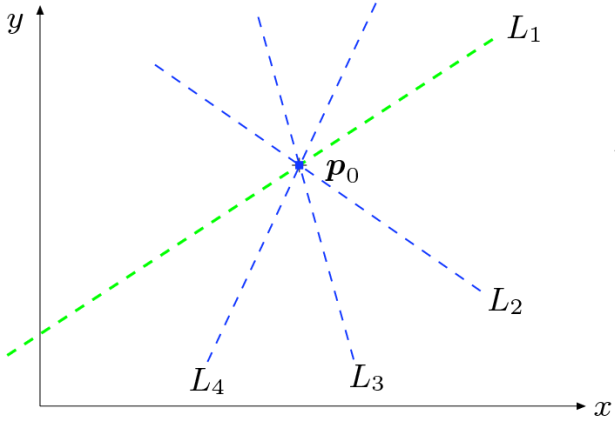
Houghova transformacija



Detekcija premic s HT



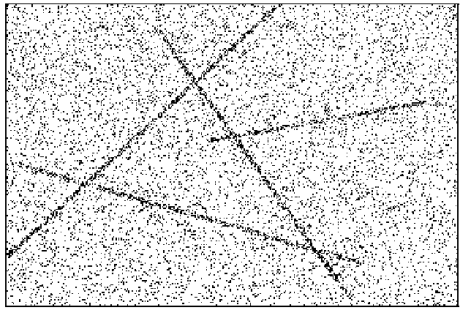
dualnost točka / premica

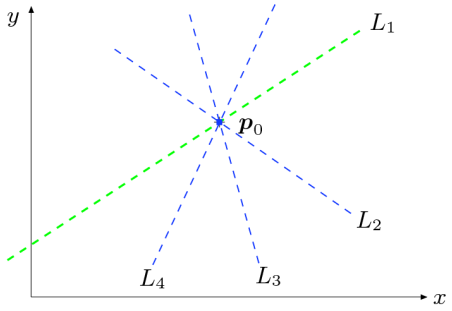


$$L_j : y_0 = k_j x_0 + d_j$$

$$d_j = -x_0 k_j + y_0$$

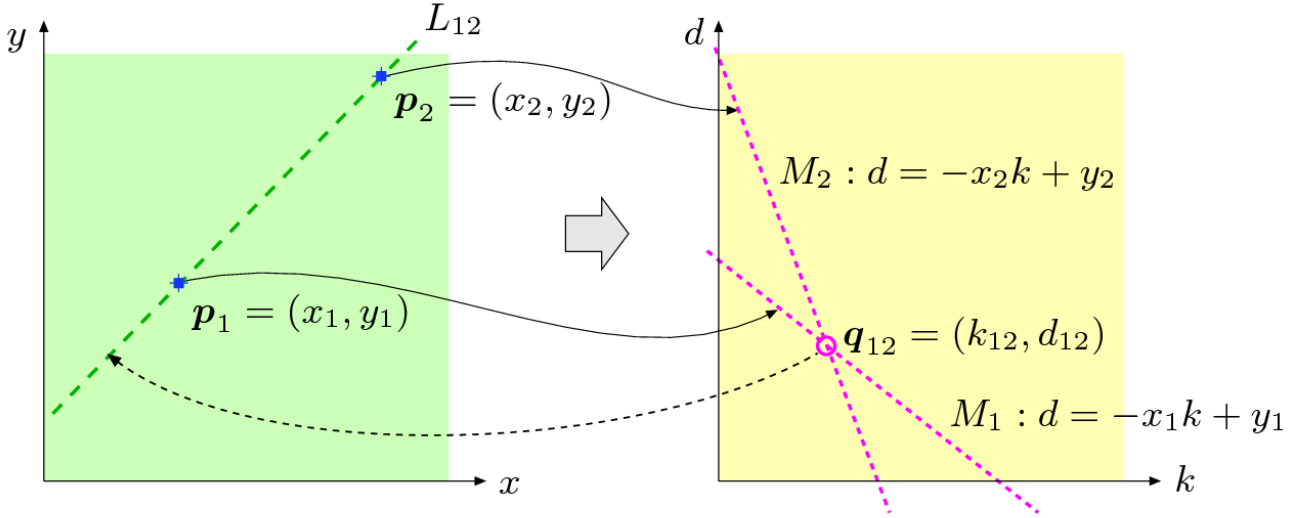
Detekcija premic s HT





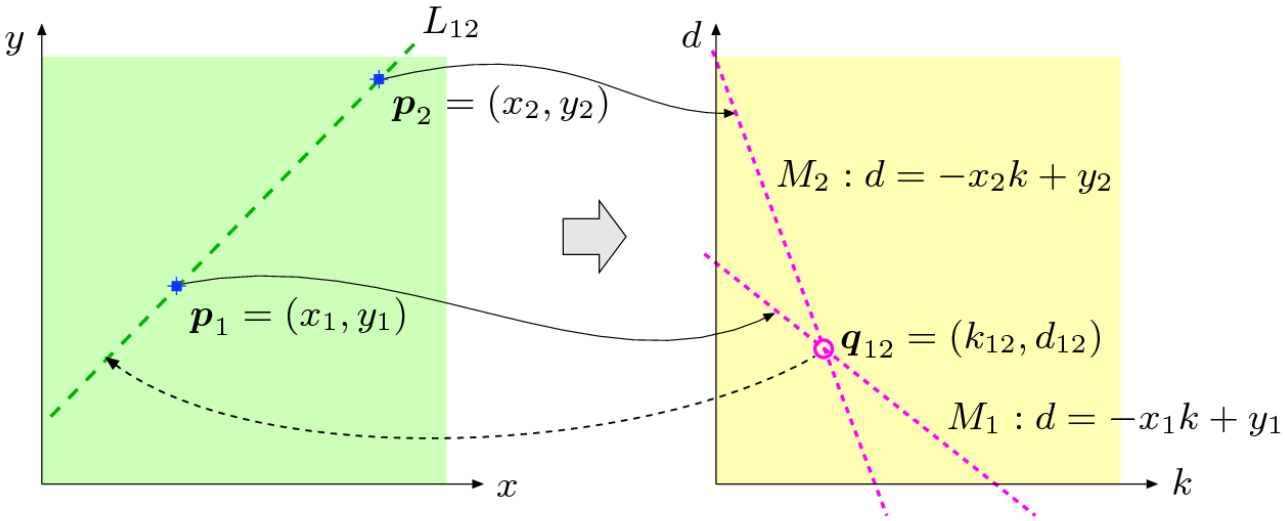
$$L_j : y_0 = k_j x_0 + d_j$$

$$d_j = -x_0 k_j + y_0$$

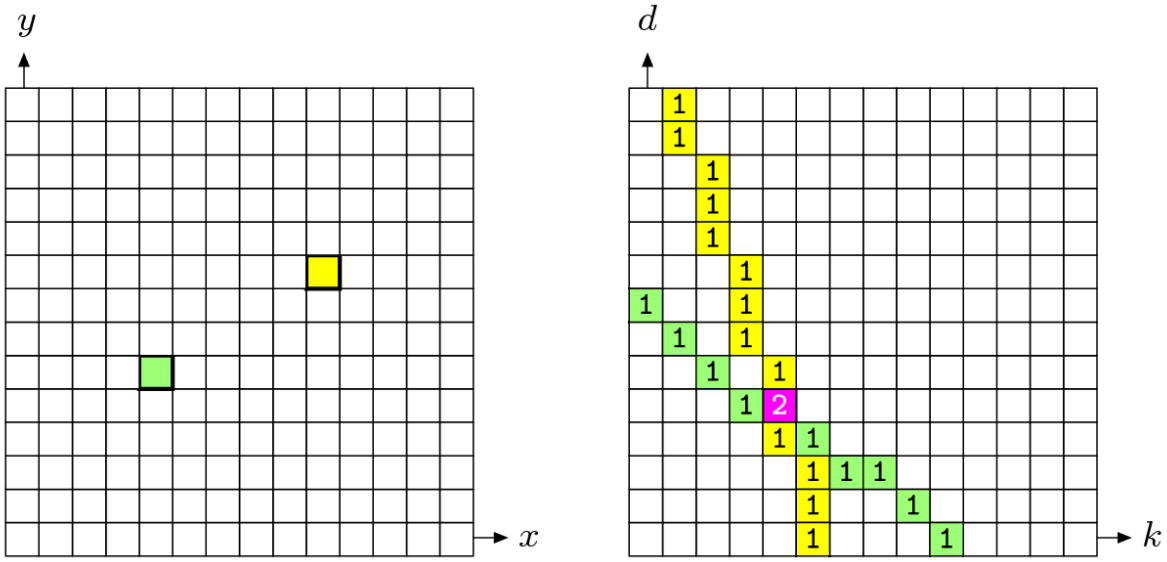


(a) x/y image space (b) k/d parameter space

Detekcija premic s HT

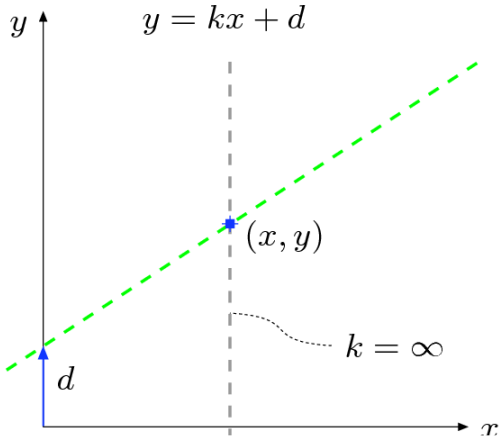


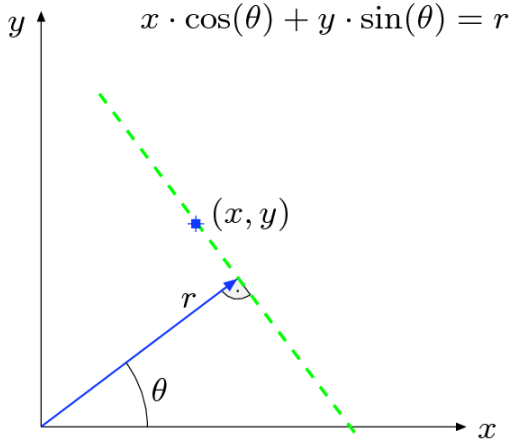
(a) x/y image space (b) k/d parameter space



(a) Image Space (b) Accumulator Array

Hessova normalna oblika enačbe premice

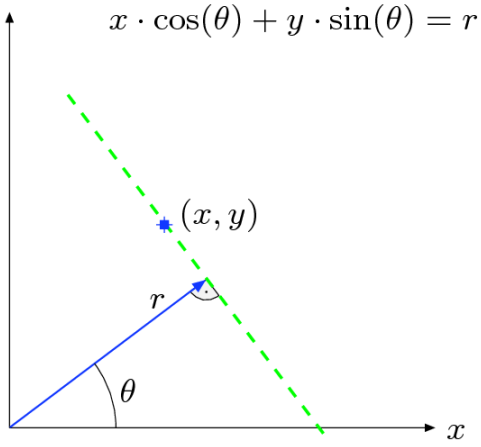




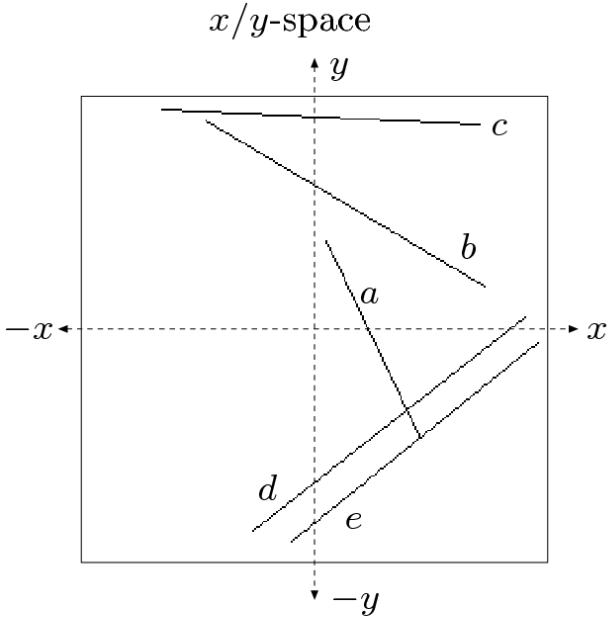
$$x \cdot \cos(\theta) + y \cdot \sin(\theta) = r$$

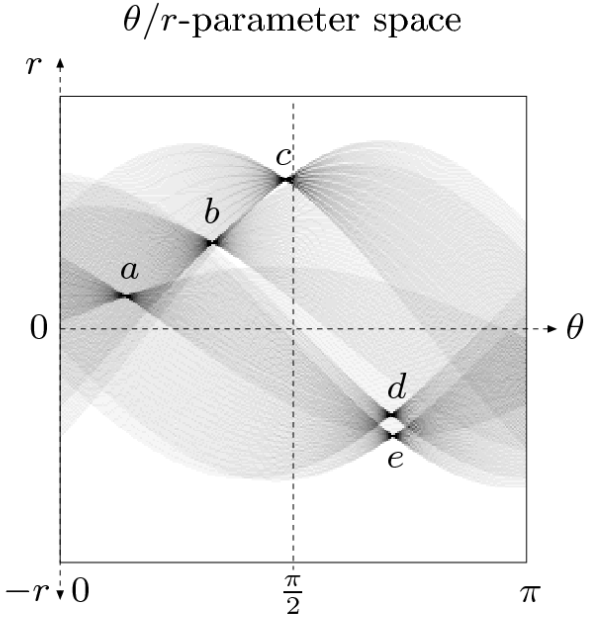
$$r_{x_i, y_i}(\theta) = x_i \cdot \cos(\theta) + y_i \cdot \sin(\theta)$$

Preslikava Hessove normalne oblike premice v parametrični prostor



$$r_{x_i,y_i}(\theta) = x_i \cdot \cos(\theta) + y_i \cdot \sin(\theta)$$





Houghova transformacija

```
1:  HoughLines(I)
        Returns the list of parameters ⟨θ_i, r_i⟩ corresponding to the strongest
        lines found in the binary image I.
2:      Set up a two-dimensional array Acc[θ, r] of counters, initialize to 0.
3:      Let (u_c, v_c) be the center coordinates of the image I
4:      for all image coordinates (u, v) do
5:          if I(u, v) is an edge point then
                Get coordinate relative to the image center (u_c, v_c):
6:              (x, y) ← (u − u_c, v − v_c)
7:              for θ_i = 0 ... π do
8:                  r_i = x · cos(θ_i) + y · sin(θ_i)
9:                  Increment Acc[θ_i, r_i]
        Return the list of parameter pairs ⟨θ_j, r_j⟩ for K strongest lines:
10:     MaxLines ← FindMaxLines(Acc, K)
11:     return MaxLines.
```

class HoughTransform

```
public class HoughTransform {

    ImageProcessor ip;
    int xCtr, yCtr;
    int nAng;
    int nRad;
    int cRad;
    double dAng;
    double dRad;
    int[][] houghArray;
    int[][] localMaxArray;

    public LinearHT(ImageProcessor ip, int nAng,
                    int nRad)     {
       this.ip = ip;
       this.xCtr = ip.getWidth()/2;
       this.yCtr = ip.getHeight()/2;
       this.nAng = nAng;
       this.dAng = Math.PI / nAng;
       this.nRad = nRad;
       this.cRad = nRad / 2;
       double rMax = Math.sqrt(xCtr * xCtr
                     + yCtr * yCtr);
       this.dRad = (2.0 * rMax) / nRad;
       this.houghArray = new int[nAng][nRad];
       fillHoughAccumulator();
       findLocalMaxima();
    }
```

...

```
    void fillHoughAccumulator() {
       int h = ip.getHeight();
       int w = ip.getWidth();
       for (int v = 0; v < h; v++) {
          for (int u = 0; u < w; u++) {
               if (ip.get(u, v) > 0)
                    pixToHough(u, v);
          }
       }
    }

    void pixToHough(int u, int v) {
       int x = u - xCtr,  y = v - yCtr;
       for (int i = 0;   i < nAng; i++) {
          double theta = dAng * i;
          int r = cRad + (int) Math.rint
                    ((x*Math.cos(theta) +
                    y*Math.sin(theta)) / dRad);
          if (r >= 0 && r < nRad)
               houghArray[i][r]++;
       }
    }
```

class HoughTransform

```
public void findLocalMaxima() {
    localMaxArray = new int[nAng][nRad];
      for (int a = 0; a < nAng; a++) {
         int a1 = (a - 1 + nAng) % nAng;
      int a2 = (a + 1) % nAng;
         for (int r = 1; r < nRad - 1; r++) {
            int ha = houghArray[a][r];
            boolean ismax =
               ha > houghArray[a1][r-1] &&
               ha > houghArray[a1][r] &&
               ha > houghArray[a1][r+1] &&
               ha > houghArray[a][r-1] &&
               ha > houghArray[a][r+1] &&
               ha > houghArray[a2][r-1] &&
               ha > houghArray[a2][r] &&
               ha > houghArray[a2][r+1] ;
            if (ismax)
               localMaxArray[a][r] = ha;
         }
      }
    }
```

Posplošen HT

Lokalne značilke modelne slike

lokacija, smer, merilo

(glede na referenčno točko v modelu)

Iščemo pojavitev modela v sliki

Modelirana funkcija je transformacija, ki model preslika v pojavitev modela v sliki