Robotika in računalniško zaznavanje (RRZ)

Iskanje robov in kotov

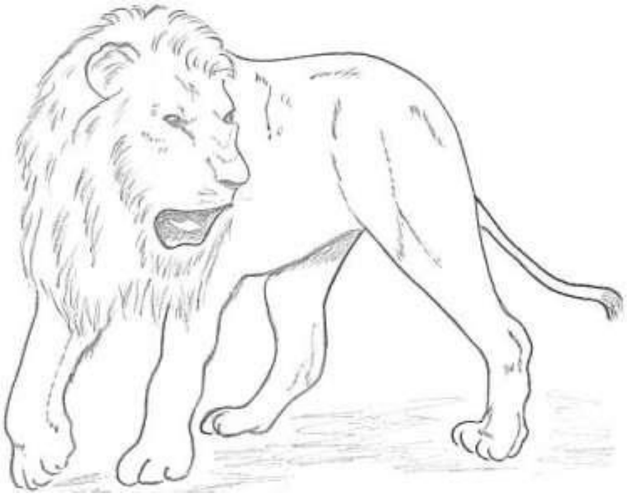
Danijel Skočaj
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

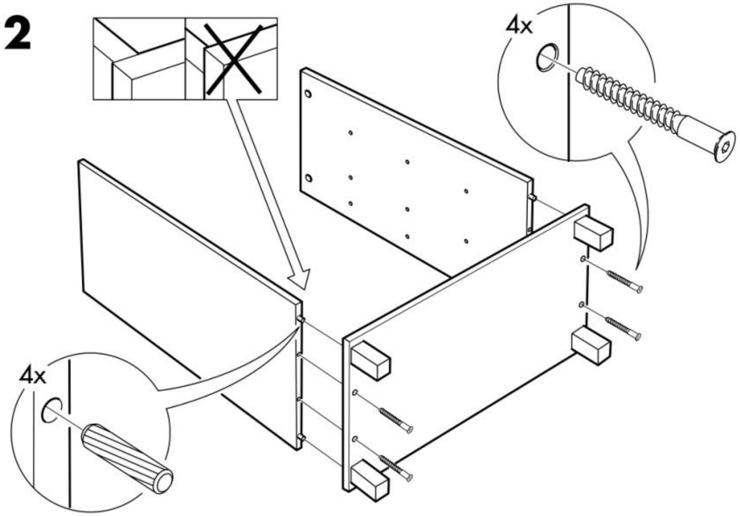
Literatura: W. Burger, M. J. Burge (2008).
Digital Image Processing, poglavja 7, 8

v7.0

Robovi

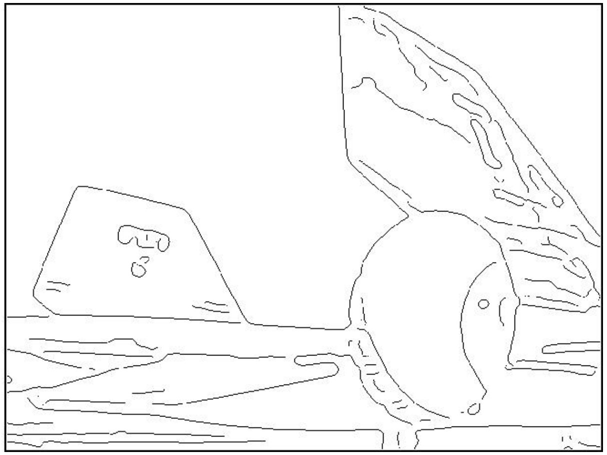
- Robovi so zelo informativni





Kaj je rob?

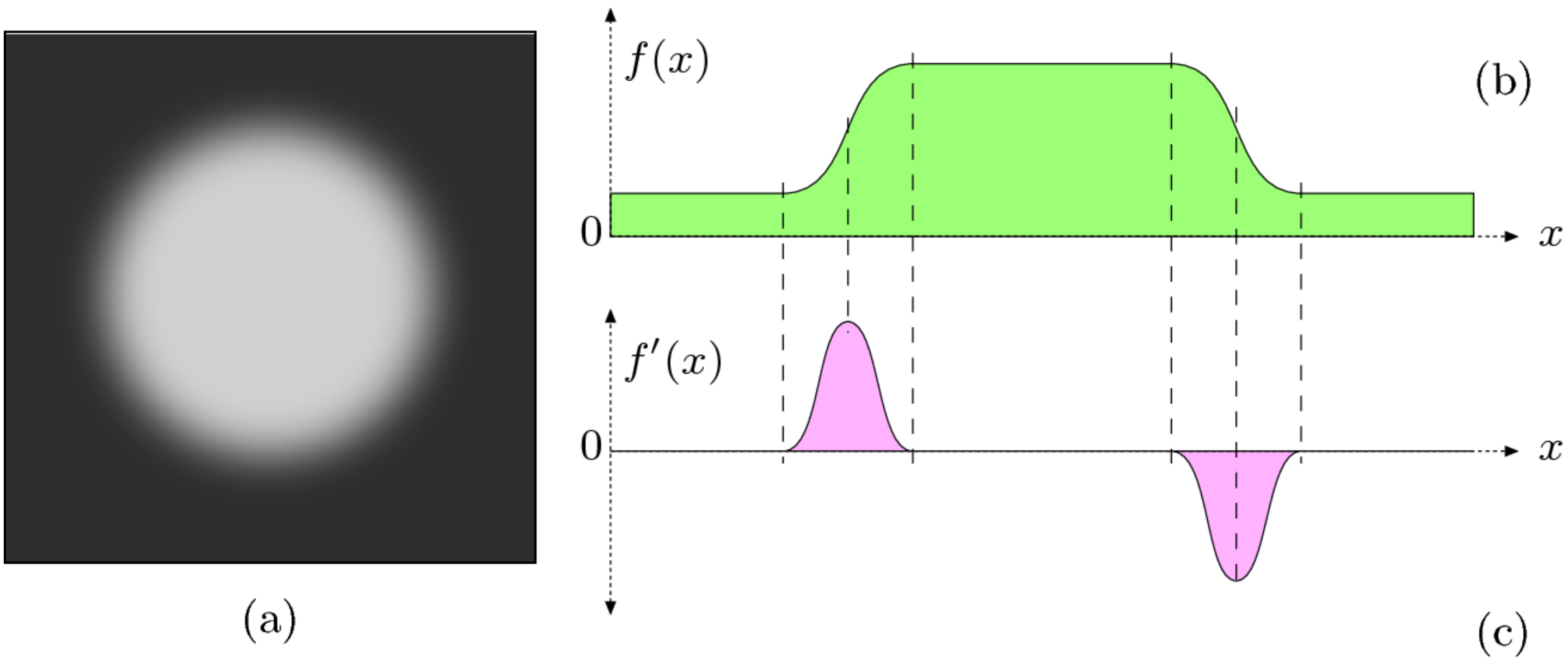
- Robovi so deli slik (slikovni elementi) kjer se lokalna intenziteta bistveno spremeni v določeni smeri



Odvod zvezne funkcije

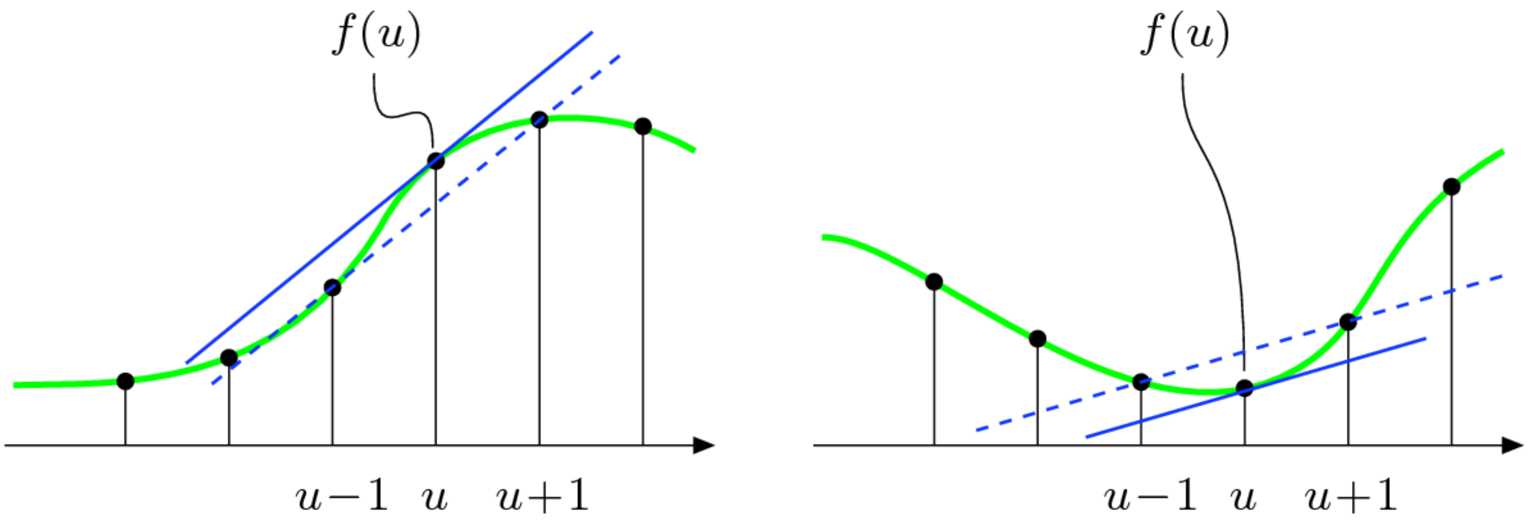
- Lokalne spremembe merimo z odvodom:

$$f'(x) = \frac{df}{dx}(x)$$



Odvod diskretne funkcije

- Aproksimacija odvoda zvezne funkcije



$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

Gradient

- Parcialni odvodi:

$$\frac{\partial I}{\partial u}(u,v) \qquad \text{and} \qquad \frac{\partial I}{\partial v}(u,v)$$

- Gradient:

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

- Magnituda gradienta:

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

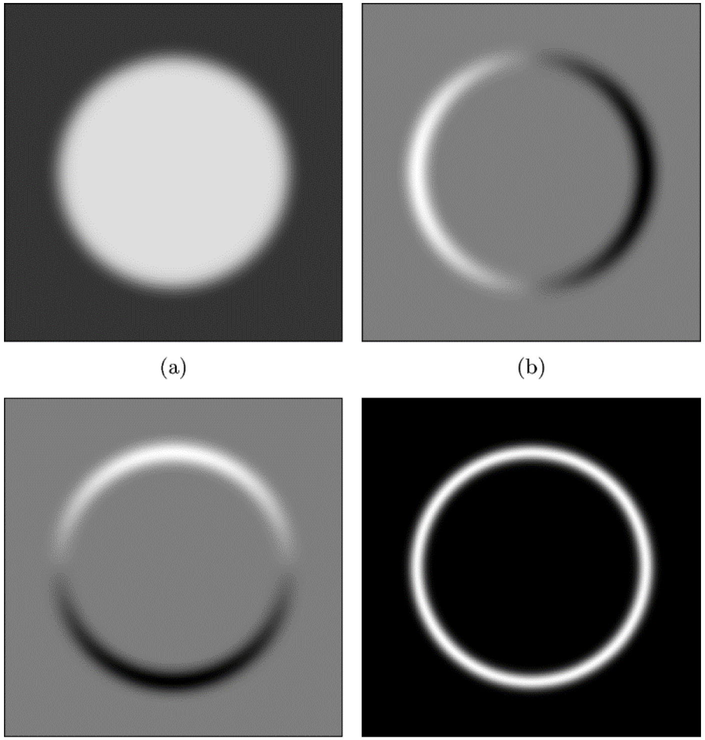
  - Rotacijsko invariantna

Filtri z odvodi

- Aproksimacijo parcialnih odvod laho računalmo z linearnimi filtri:

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$

$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$



(a) (b) (c) (d)

Operatorji za iskanje robov

- Večina operatorjev za iskanje robov temelji na aproksimaciji lokalnega gradienta
  - Moč roba
  - Orientacija roba

- Različni (podobni) operatorji za iskanje robov
  - Prewittov in Sobelov operator
  - Robertsov operator
  - Operator Kompas
  - Cannyjev operator

Prewittov operator

- Pred računanjem gradienta še malce zgladi okolico

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H_x^P = \begin{bmatrix} 1 \\ \mathbf{1} \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} 1 & \mathbf{1} & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

- Gradient:

$$\nabla I(u,v) \approx \frac{1}{6} \cdot \begin{bmatrix} (I * H_x^P)(u,v) \\ (I * H_y^P)(u,v) \end{bmatrix}$$

Sobelov operator

- Več poudarka na trenutno srednjo vrstico oz. stolpec pri glajenju:

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Gradient:

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{bmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{bmatrix}$$

Moč in orientacija robov

- Rezultat filtriranja:

$$D_x(u,v) = H_x * I \quad \text{and} \quad D_y(u,v) = H_y * I$$

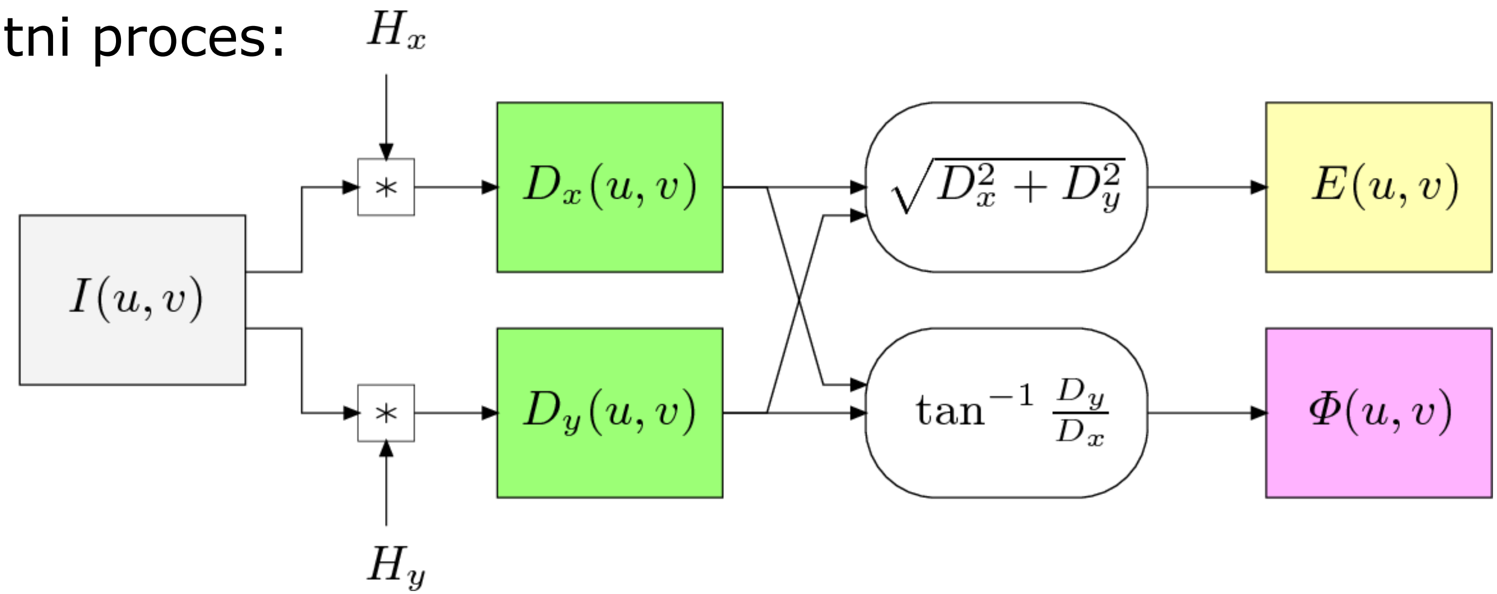
- Lokalna moč roba = magnituda gradienta

$$E(u,v) = \sqrt{\left(D_x(u,v)\right)^2 + \left(D_y(u,v)\right)^2}$$

- Lokalna orientacija roba:

$$\Phi(u,v) = \tan^{-1}\left(\frac{D_y(u,v)}{D_x(u,v)}\right) = \mathrm{ArcTan}\left(D_x(u,v), D_y(u,v)\right)$$

- Celotni proces:



Izboljšani Sobelov operator

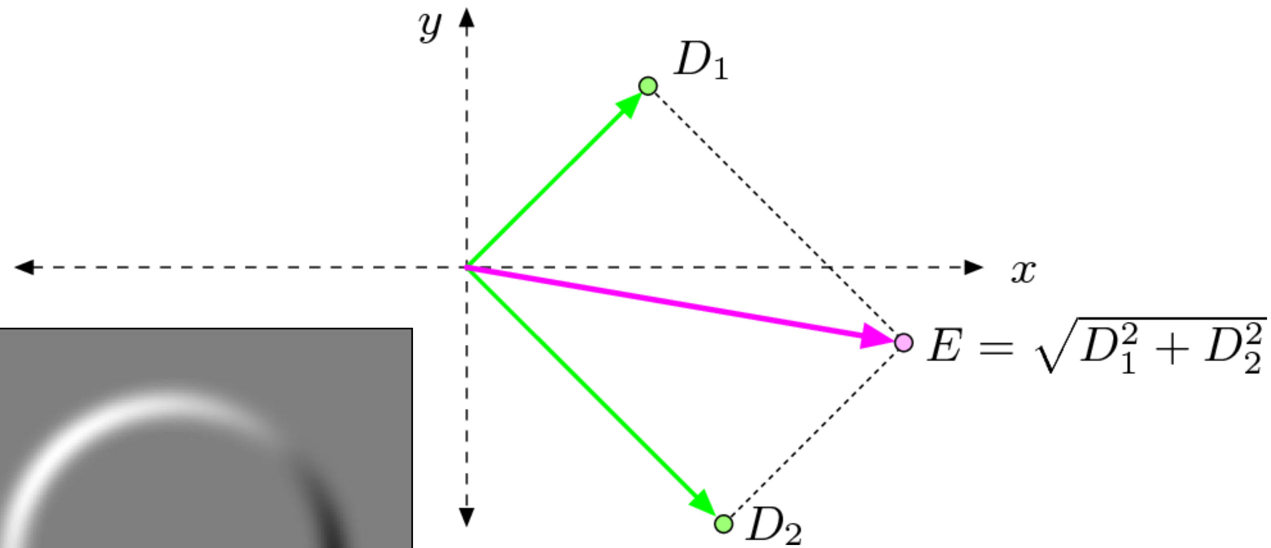
- Originalna Prewittov in Sobelov operator slabo ocenita orientacijo robov
- Izboljšani Sobelov operator:

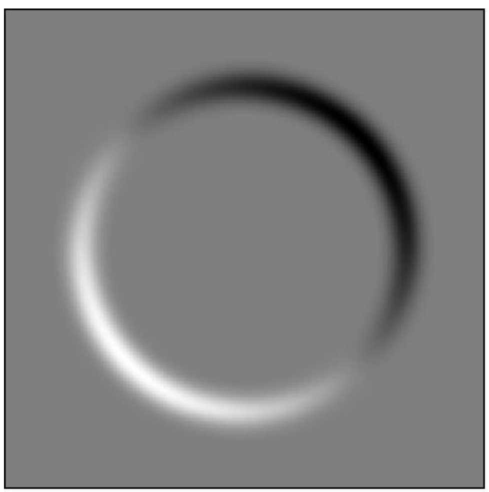
$$H_x^{S'} = \frac{1}{32}\begin{bmatrix} -3 & 0 & 3 \\ -10 & \mathbf{0} & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad \text{and} \quad H_y^{S'} = \frac{1}{32}\begin{bmatrix} -3 & -10 & -3 \\ 0 & \mathbf{0} & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

Robertsov operator

- Zelo enostaven in star

$$H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$





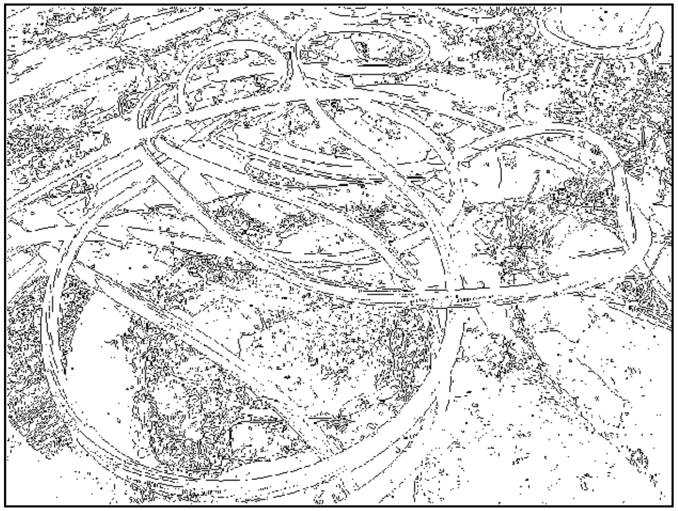
$D_1 = I * H_1^R$

$D_2 = I * H_2^R$

Primerjava različnih operatorjev

Original

Roberts



Prewitt

Sobel

Operator Kompas

- Filtri prilagojeni osmim orientacijam

$$H_0^K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_4^K = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$H_1^K = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad H_5^K = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$H_2^K = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_6^K = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$H_3^K = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \qquad H_7^K = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

Operator Kompas

- Aplicirati je potrebno samo štiri filtre:

$$I * H_4^K \; = \; I * -H_0^K \; = \; -(I * H_0^K)$$

$$\begin{array}{llll} D_0 \leftarrow I * H_0^K & D_1 \leftarrow I * H_1^K & D_2 \leftarrow I * H_2^K & D_3 \leftarrow I * H_3^K \\ D_4 \leftarrow -D_0 & D_5 \leftarrow -D_1 & D_6 \leftarrow -D_2 & D_7 \leftarrow -D_3 \end{array}$$

- Moč roba = maksimalnemu odzivu

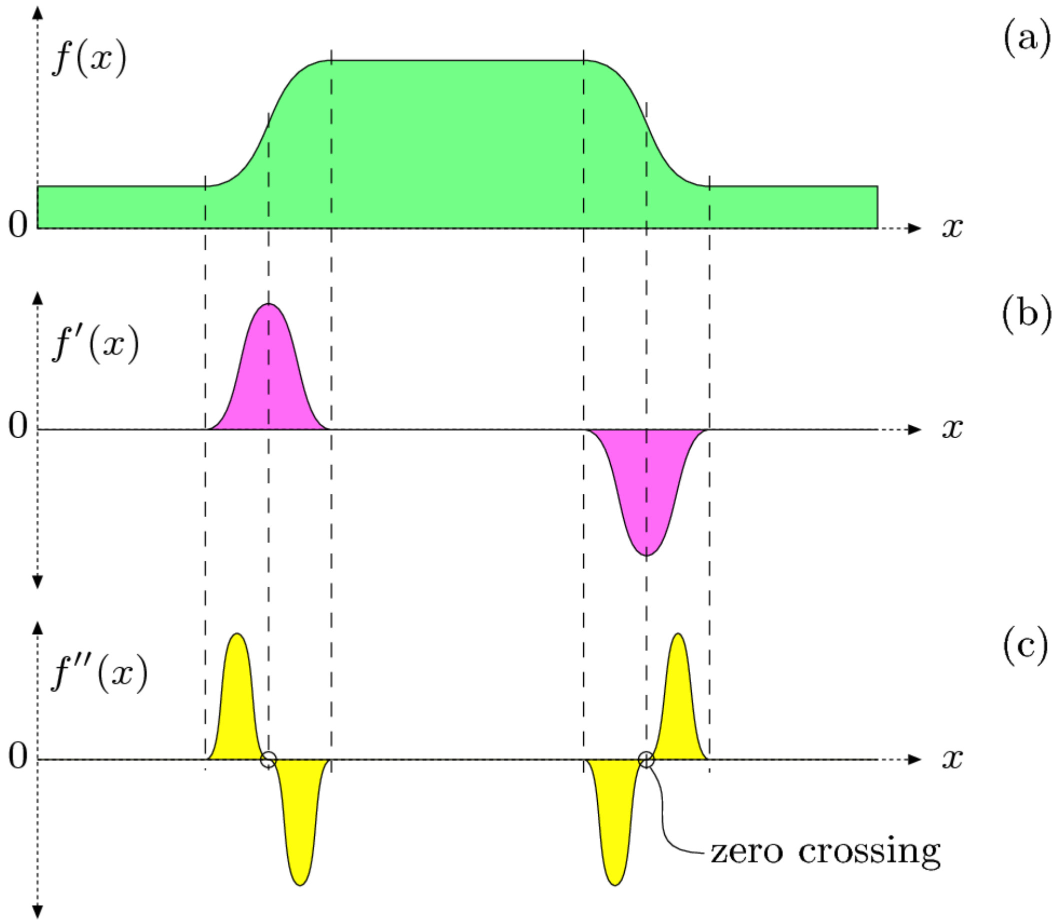
$$\begin{aligned} E^K(u,v) &\triangleq \max\big(D_0(u,v), D_1(u,v), \ldots D_7(u,v)\big) \\ &= \max\big(|D_0(u,v)|, |D_1(u,v)|, |D_2(u,v)|, |D_3(u,v)|\big) \end{aligned}$$

- Orientacija roba = orientaciji filtra z maksimalnim odzivom:

$$\Phi^K(u,v) \triangleq \frac{\pi}{4} \qquad \text{with} \;\; j = \operatorname*{argmax}_{0 \le i \le 7} \; D_i(u,v)$$

Drugi odvodi za detekcijo robov

- Včasih je rob težko lokalizirati na osnovi prvega odvoda
- Išče se ničla drugega odvoda



- Primer: LoG operator (Laplacian-of-Gaussian)

Robovi na različnih skalah

- Človek gleda robove ne samo v zelo majhni okolici ampak v večjem kontekstu
- Človek gleda robove na večih skalah

=> multiresolucijske tehnike za iskanje robov, ki iščejo robove na večih skalah

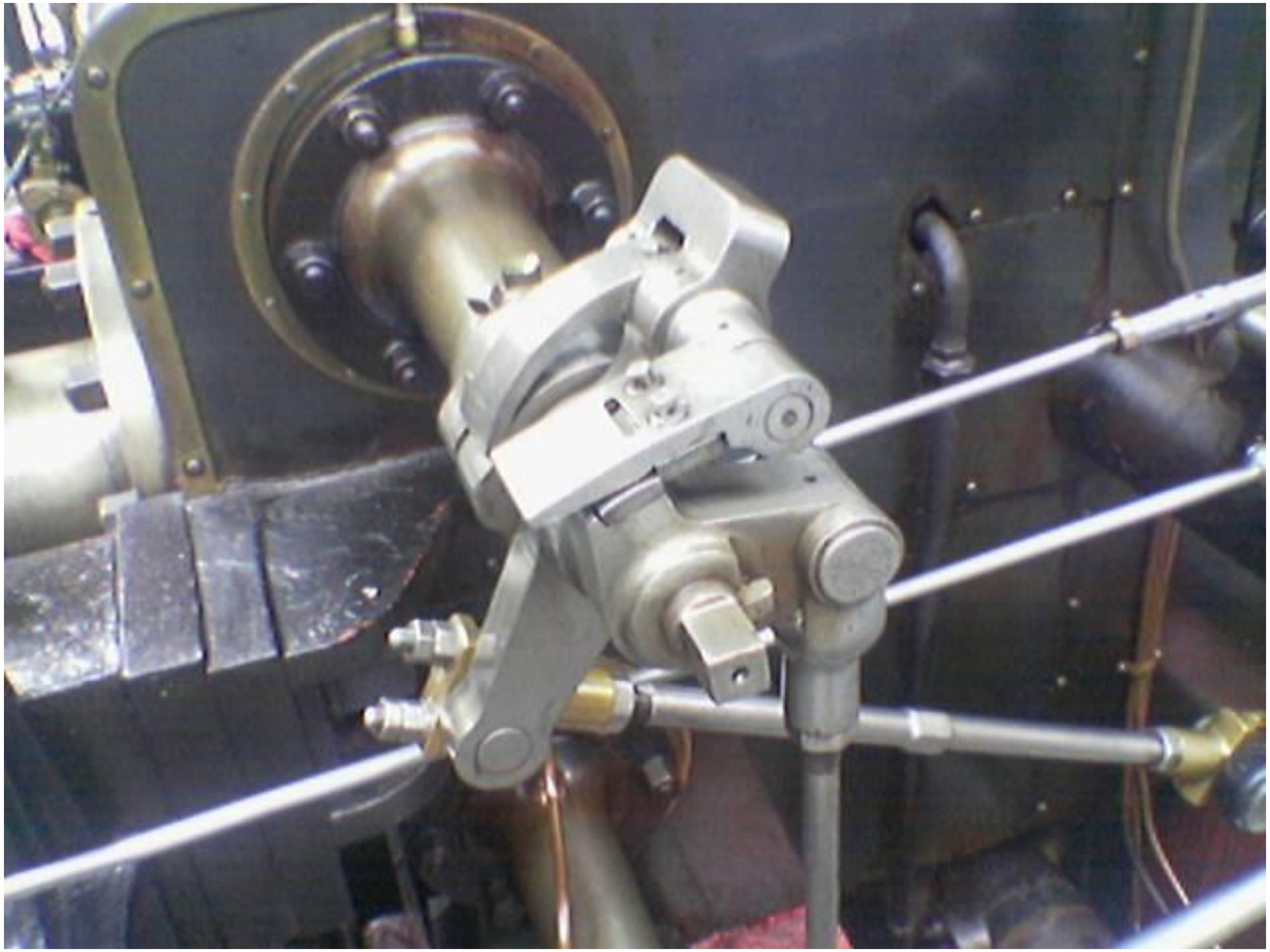
Cannyjev operator

- Najbolj popularen operator
- Tri cilji:
  - Minimizirati število napačno detektiranih robnih točk
  - Dobra lokalizacija robov
  - Tanki robovi
- Uporablja prve odvode za detekcijo robov
- Uporablja druge odvode za tanjšanje robov (natančno lokalizacijo robov)
- Deluje na večih skalah (resolucijah slike)
  - Ali na eni skali s filtrom s prilagodljivim premerom
- Ponavadi daje najboljše rezultate

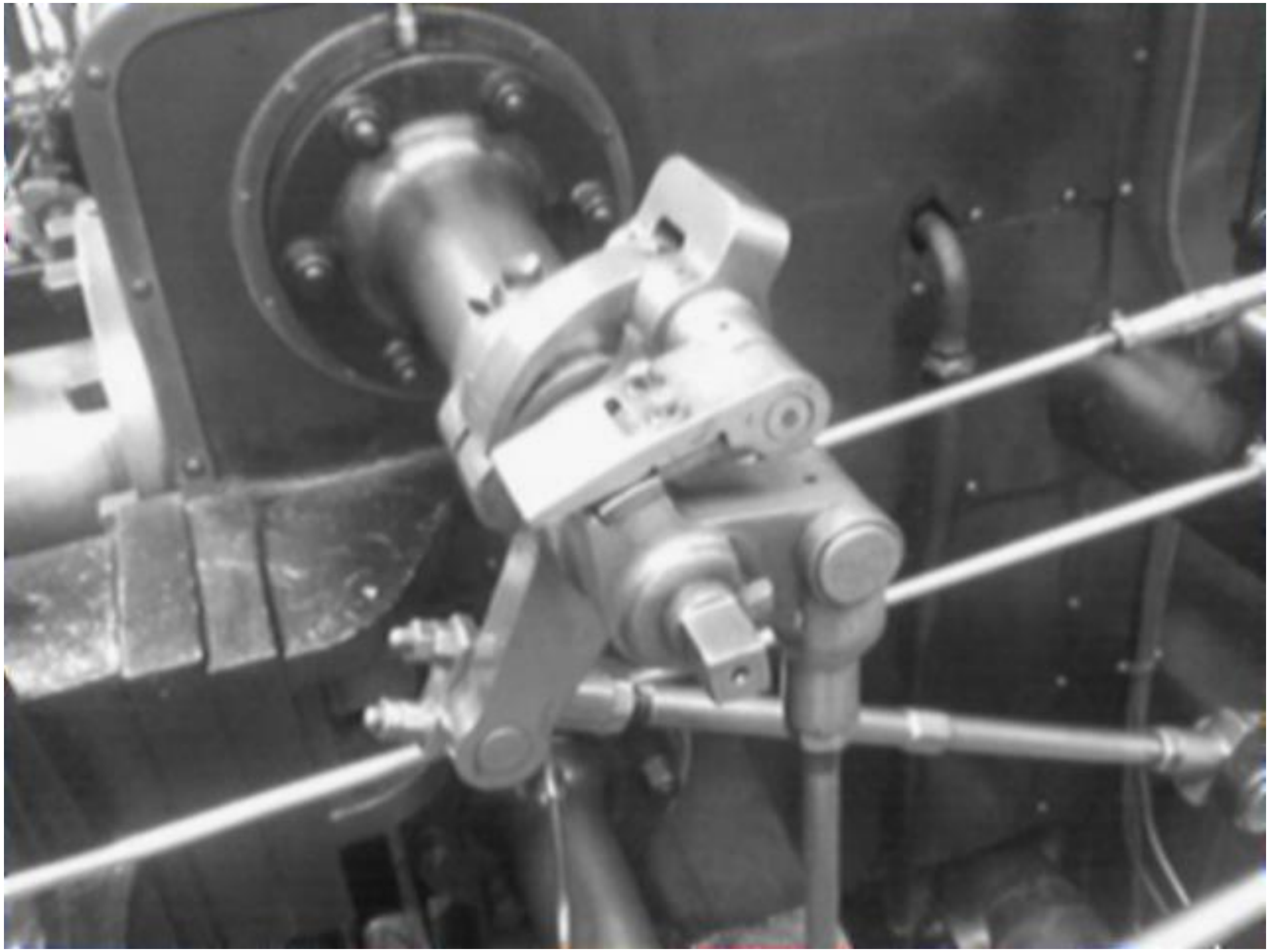
Algoritem

1. Zmanjševanje šuma
   - Glajenje z Gaussovim filtrom
2. Računanje gradienta
   - Aplicira se linearni gradientni filter za iskanje robov (npr. Sobelov)
   - Izračuna se magnituda in smer roba (ki se zaokroži na enega od štirih kotov (0, 45, 90 in 135 stopinj))
   - Magnitude se upragovi, da dobimo sliko robov
3. Dušenje lokalnih ne-maximumov
   - Tanjšanje robov
   - Preveri velikosti magnitude dveh sosednjih elementov v smeri gradienta; če vrednost trenutnega slikovnega elementa ni največja, jo postavi na nič
4. Sledenje robov z uporagovljenjem na osnovi histerez
   - Rob se začne slediti, če je magnituda večja od T1 in se sledi vse dokler ni manjša od T2

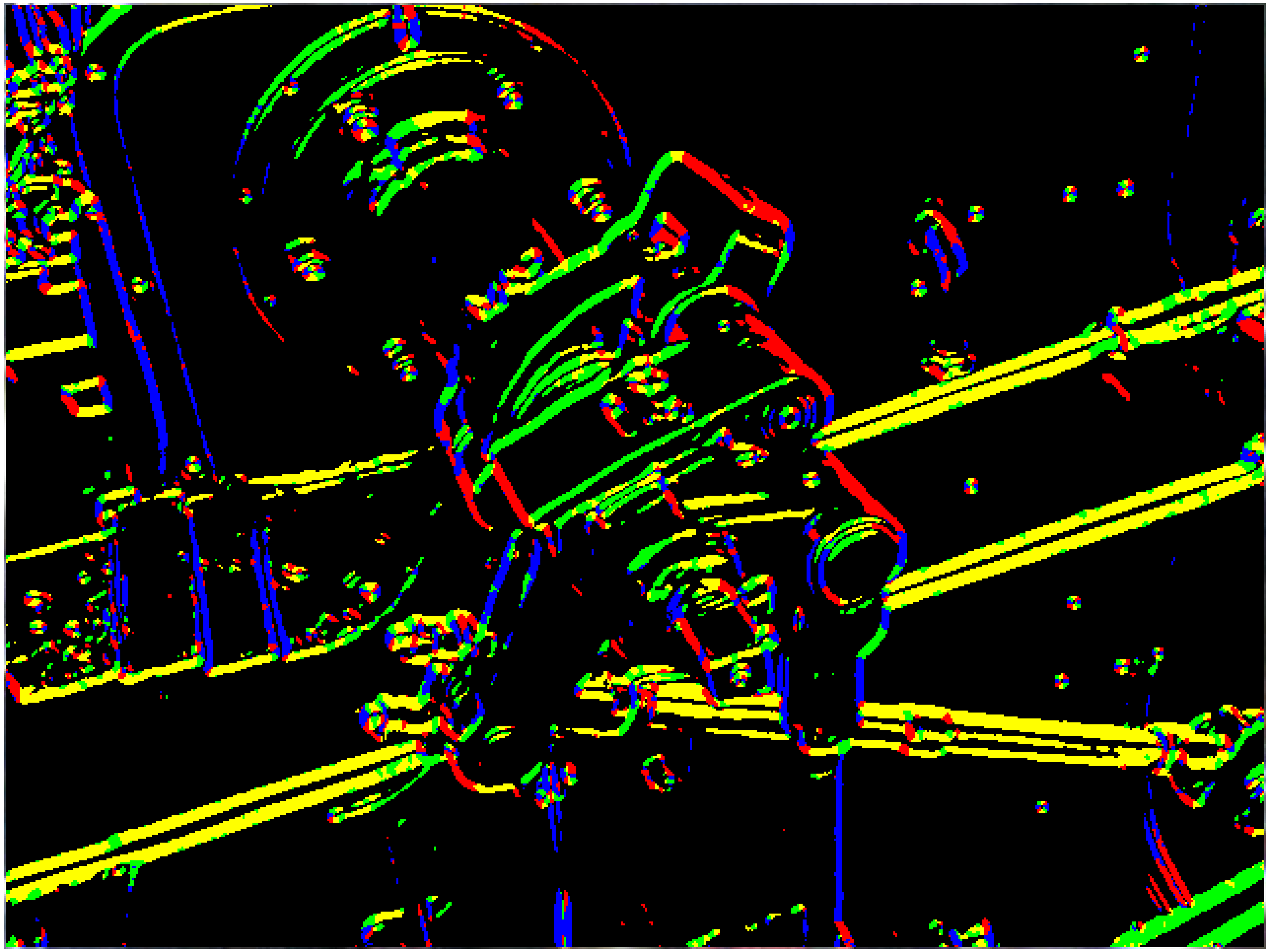
Primer – originalna slika



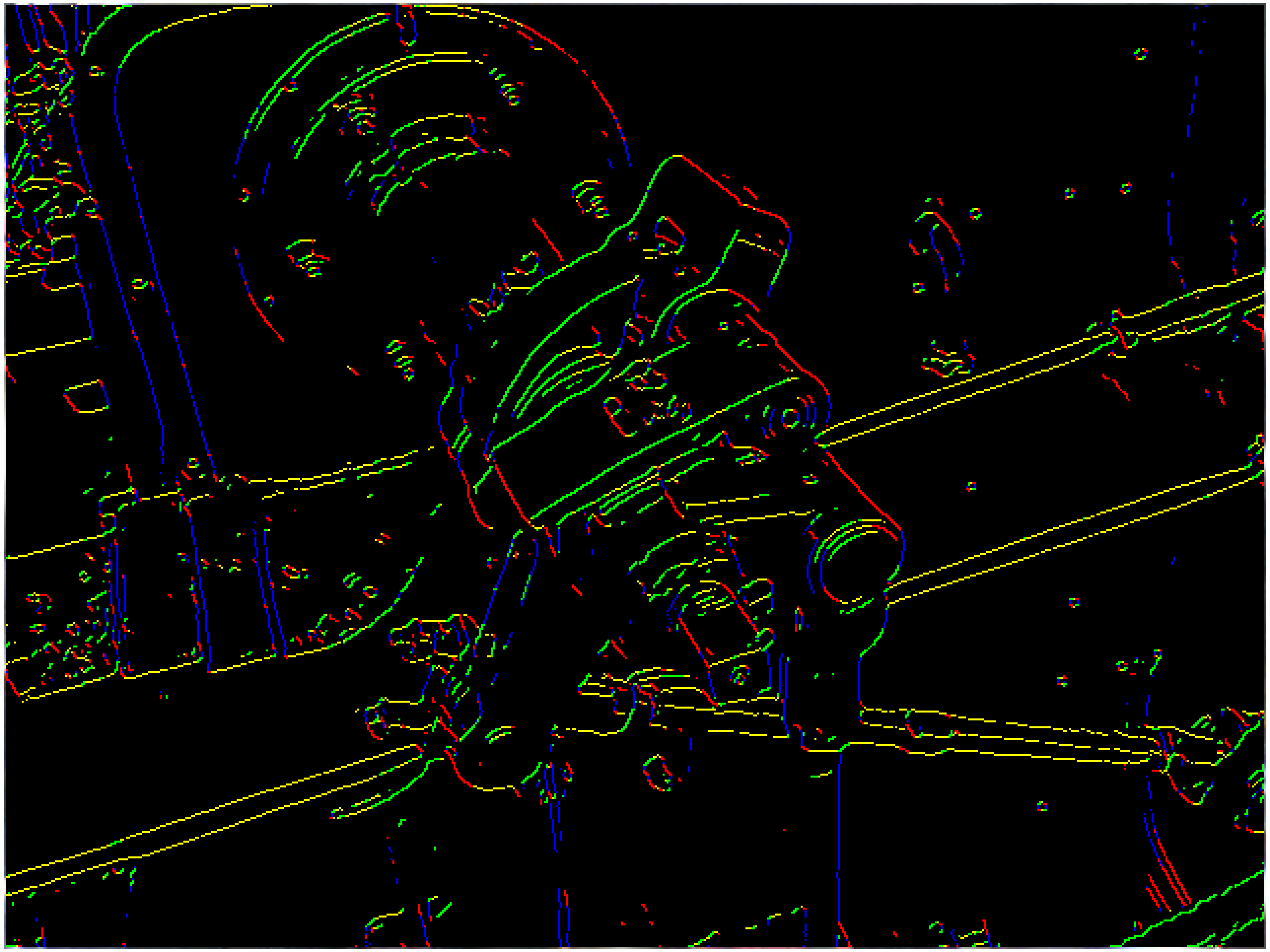
Primer – zglajena slika



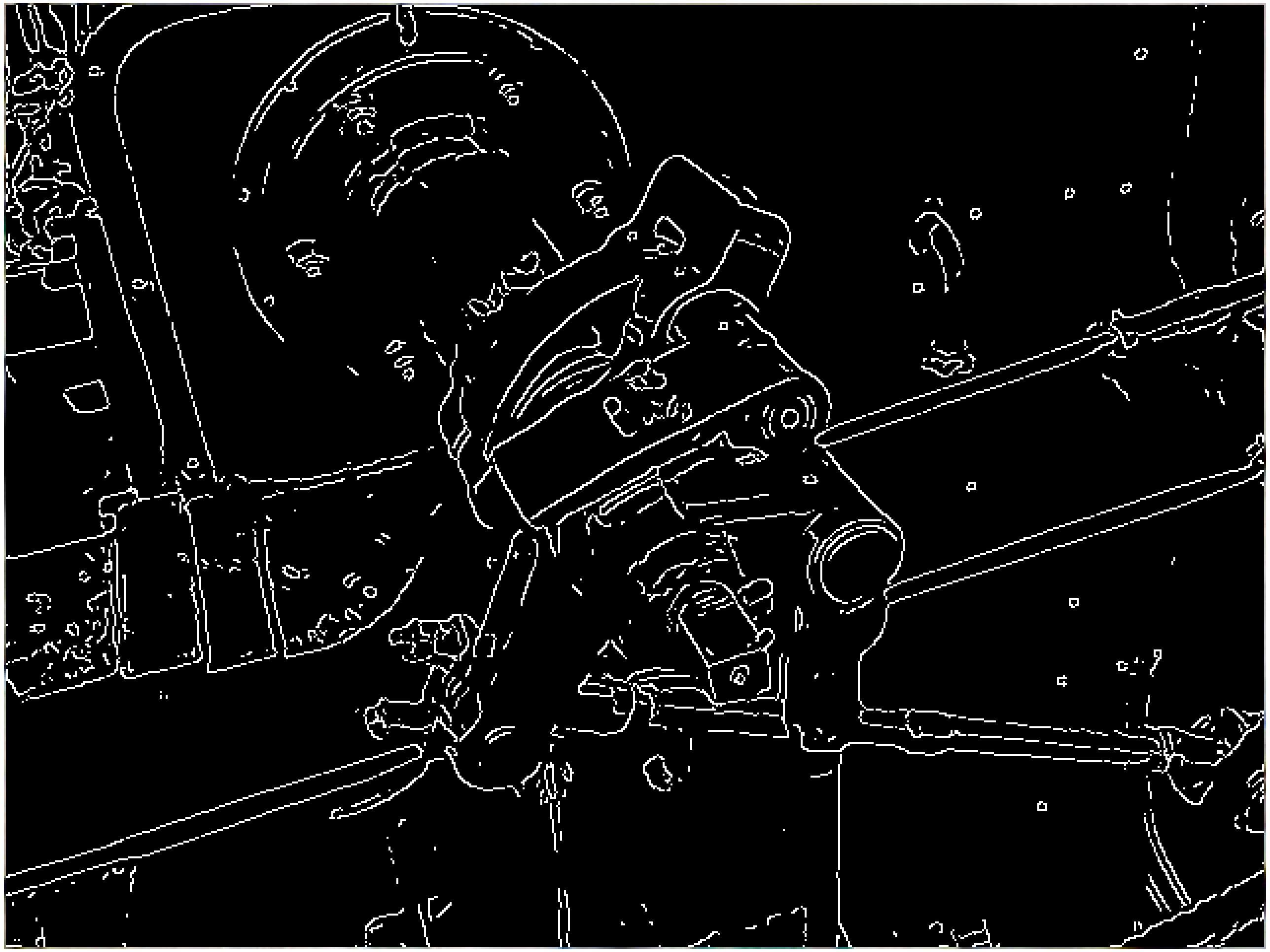
Primer – slika robov



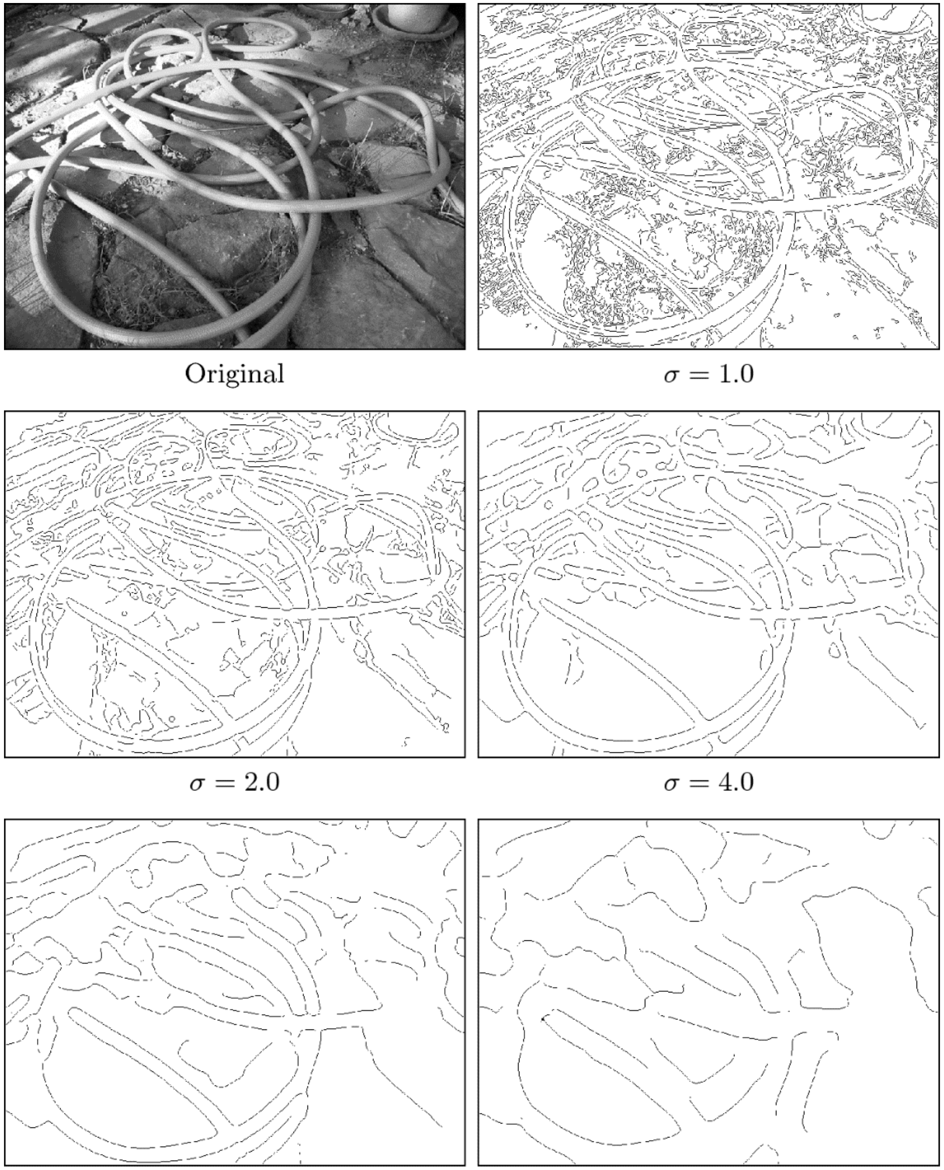
Primer – slika tankih robov



Primer – končni rezultat

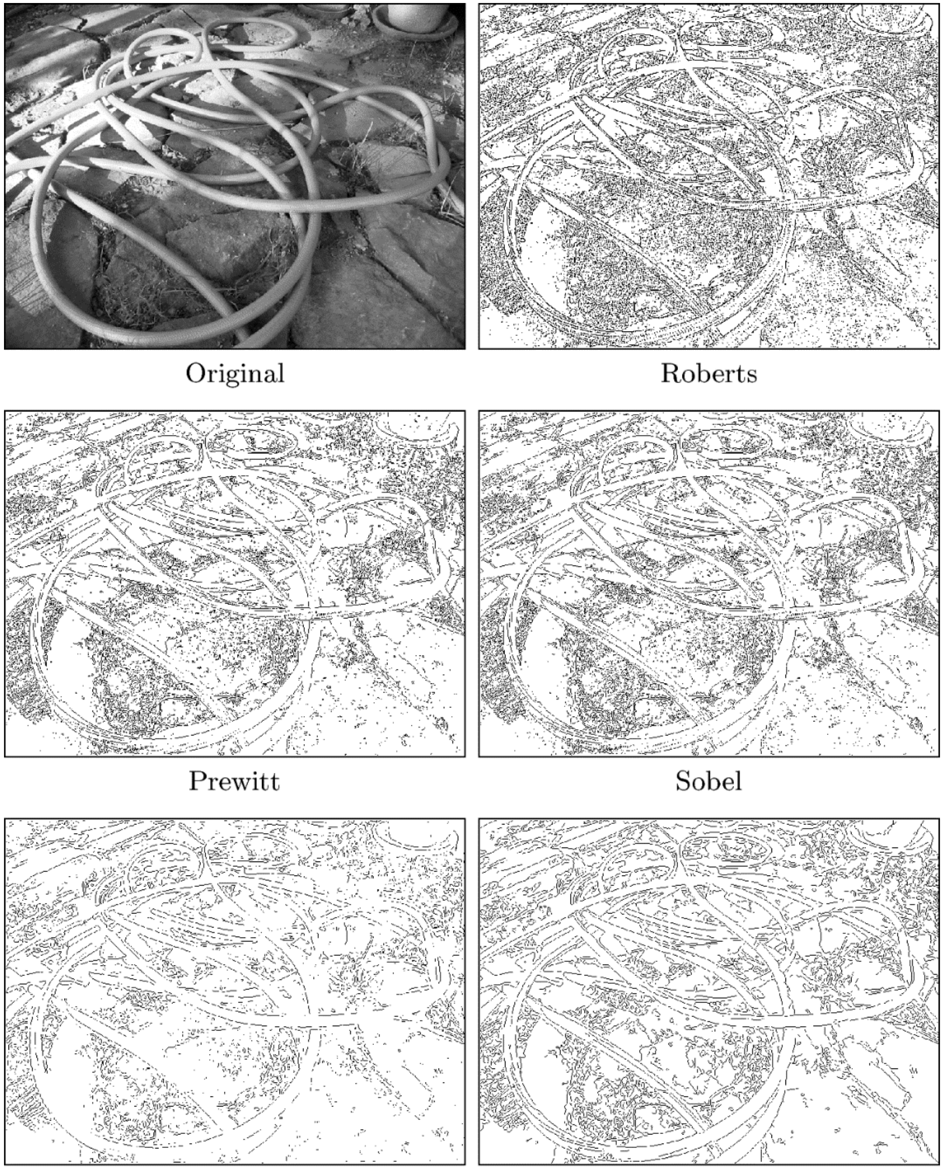


Rezultati Cannyjevega operatorja



Original; $\sigma = 1.0$; $\sigma = 2.0$; $\sigma = 4.0$; $\sigma = 8.0$; $\sigma = 16.0$

Primerjava različnih operatorjev



Original | Roberts

Prewitt | Sobel

Laplacian of Gaussian | Canny ($\sigma = 1.0$)

Od robov do obrisov

- Enostavno sledenje robovom ponavadi ne deluje
  - Robovi se včasih končajo v regijah z majhnim gradientom
  - Robovi se sekajo
  - Obrisi se lahko vejajo v več smeri
- Lažje v enostavnih primerih in v primeru binarnih slik

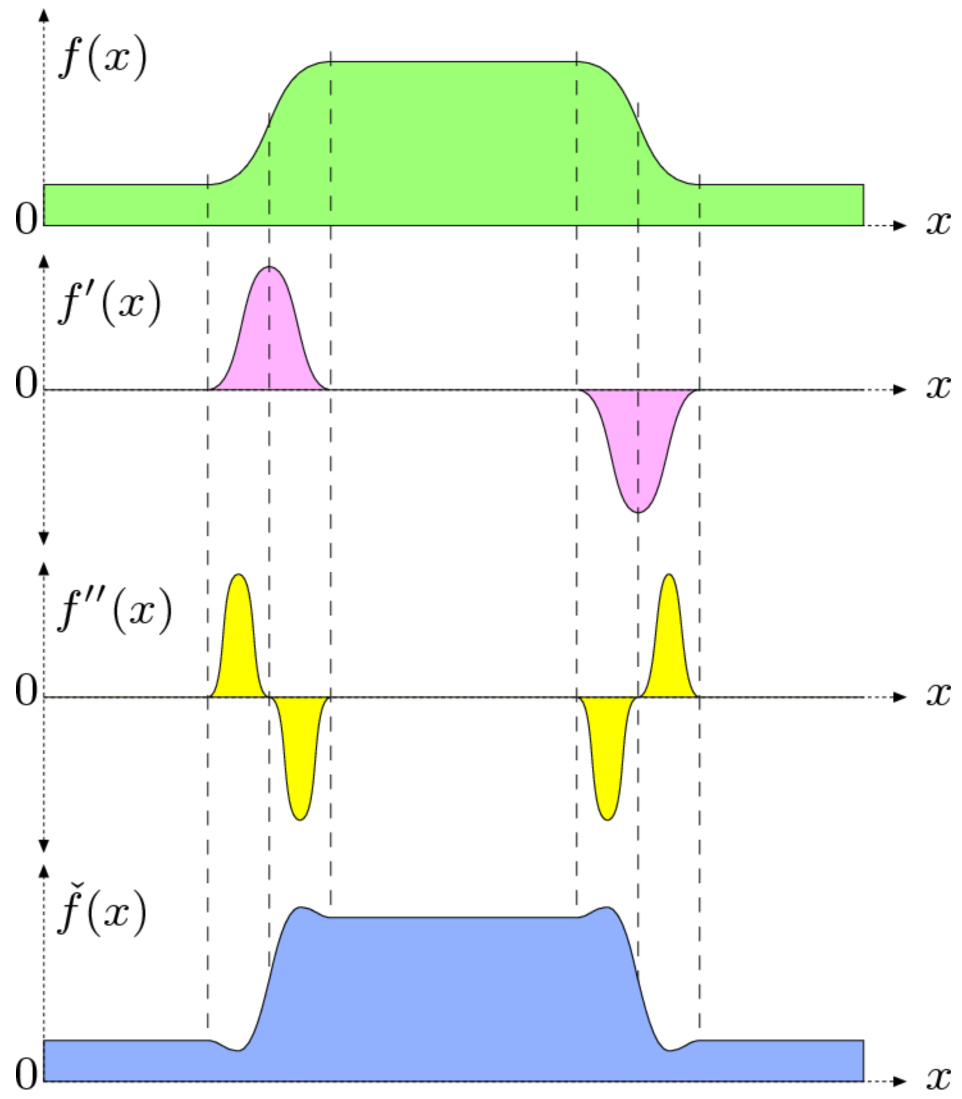
- Zemljevid robov
  - Če je izhod operatorja za iskanje robov sivinska slika, jo upragovimo in dobimo sliko robov (edge map)
    - Globalno upragovljenje
      - Ne nejboljše, prekinjene črte, ipd.
      - Dovolj dobro za nekatere vrste nadaljnjega procesiranja (Houghova transformacija)
    - Lokalno upragovljenje
      - Na osnovi histerez (Canny)

Ostrenje robov

- Robove lahko poudarimo -> poudarimo visoko-frekvenčne dele slike

- Odštejemo skalirano vrednost drugega odvoda

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



Laplaceov operator

- Laplaceov operator je vsota vrednosti drugih parcialnih odvodov:

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

- Računamo ga z linearnimi filtri:

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

- Oz, ker $I * H^L = I * (H_x^L + H_y^L) = (I * H_x^L) + (I * H_y^L)$

- je Laplaceov filter:

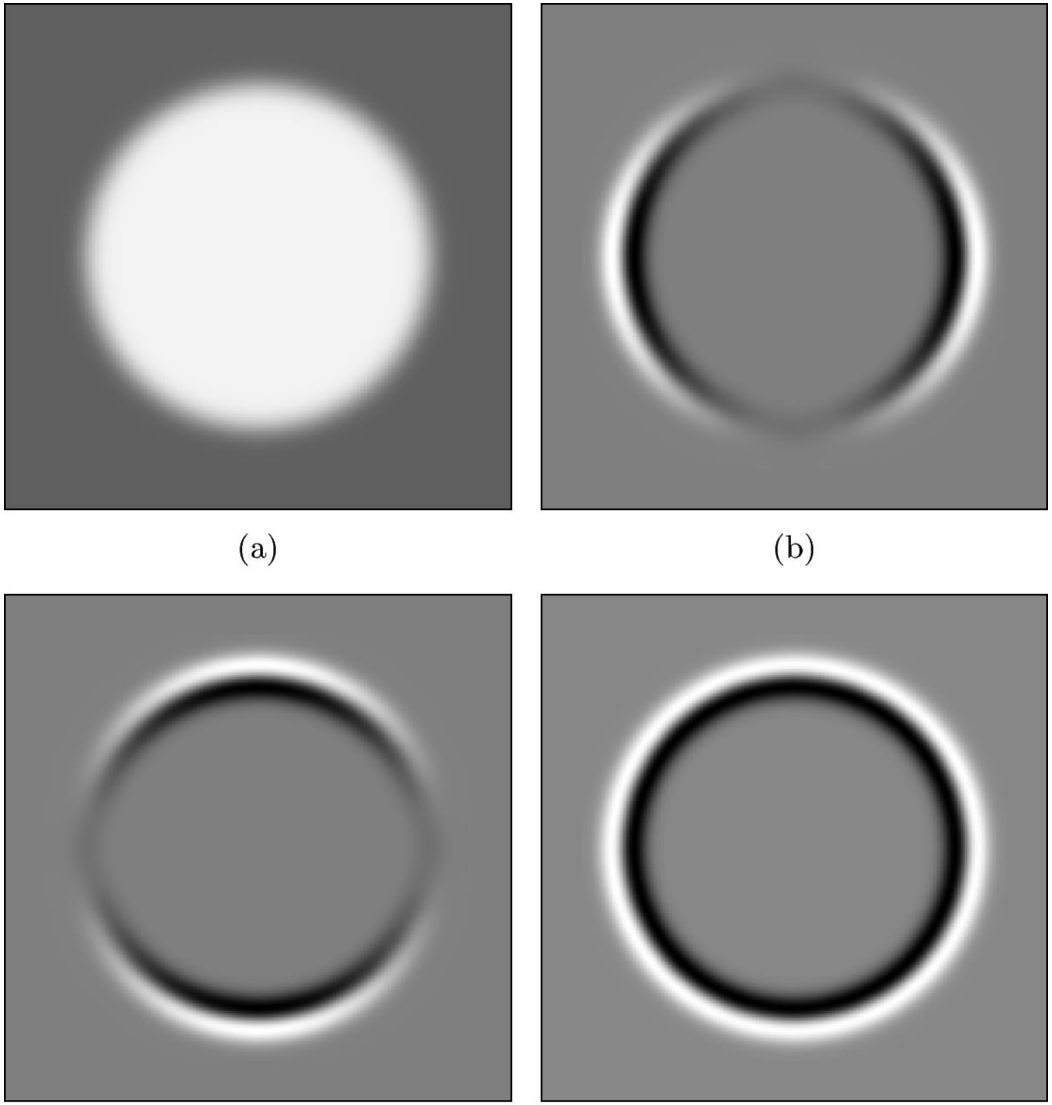
$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$H_8^L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -\mathbf{8} & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

oziroma:

$$H_{12}^L = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -\mathbf{12} & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

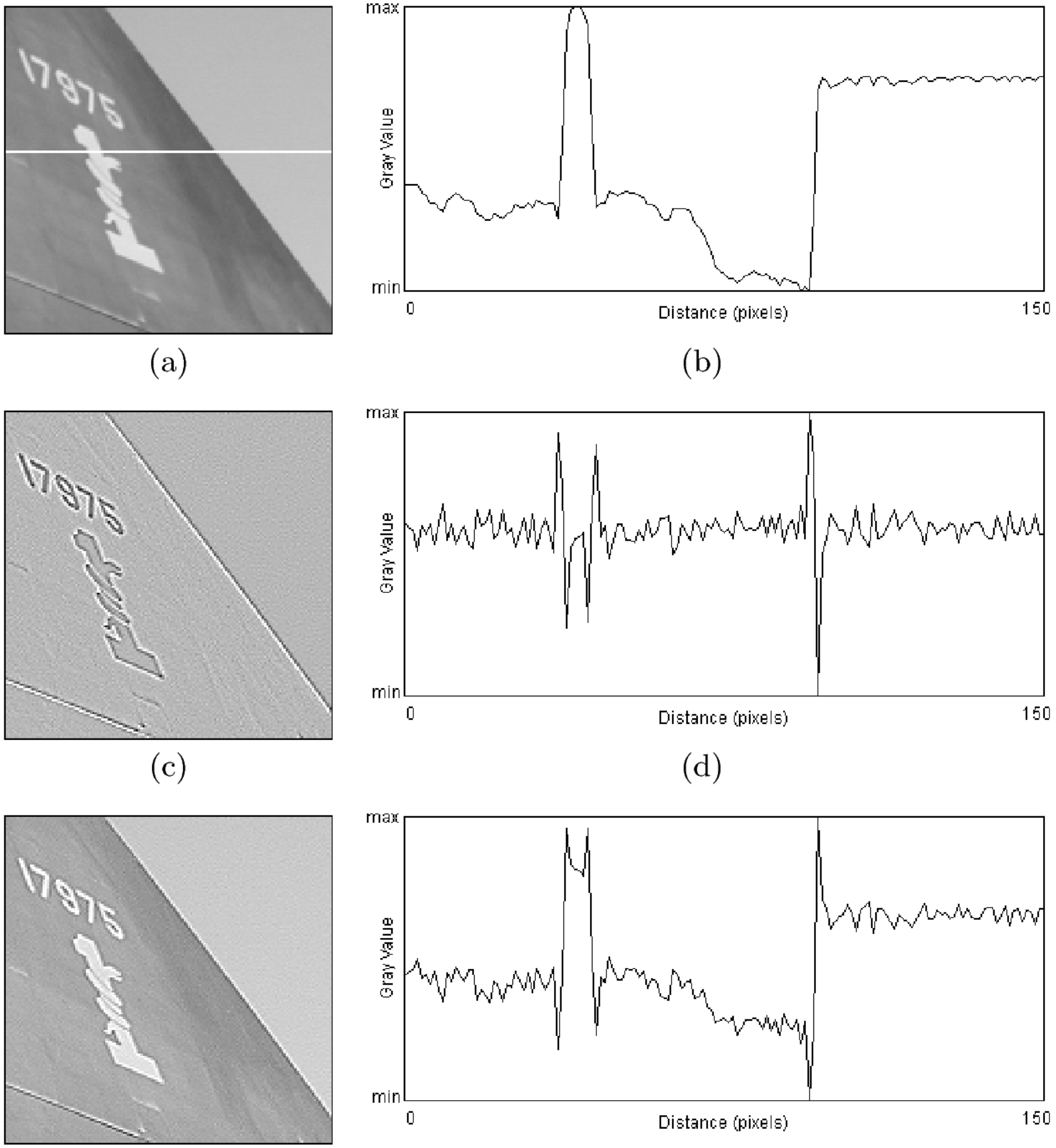
Učinek Laplaceovega operatorja


(a) (b) (c) (d)

Ostrenje

- Laplaceov operator + odštevanje od originalne slike:

$$\check{I} \leftarrow I - w \cdot (H^L * I)$$



(a) (b) (c) (d) (e) (f)

Ostrenje z neostro masko (USM filter)

- Znano iz analogne fotografije:
  1. Generiraj neostro masko (z Gaussovim filtrom):

$$M \leftarrow I - (I * \tilde{H}) = I - \tilde{I}$$

  2. Prištej skalirano masko originalni sliki:

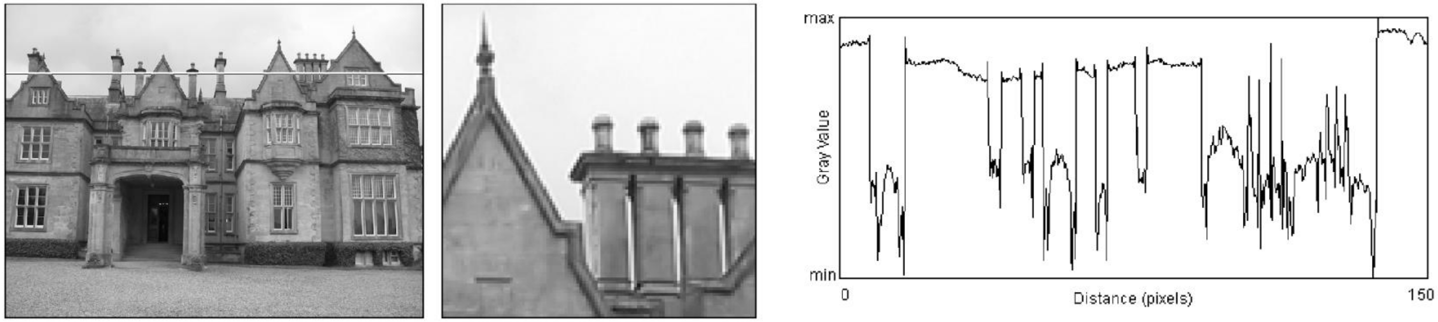
$$\check{I} \leftarrow I + a \cdot M$$

$$\check{I} \leftarrow I + a \cdot (I - \tilde{I}) = (1 + a) \cdot I - a \cdot \tilde{I}$$

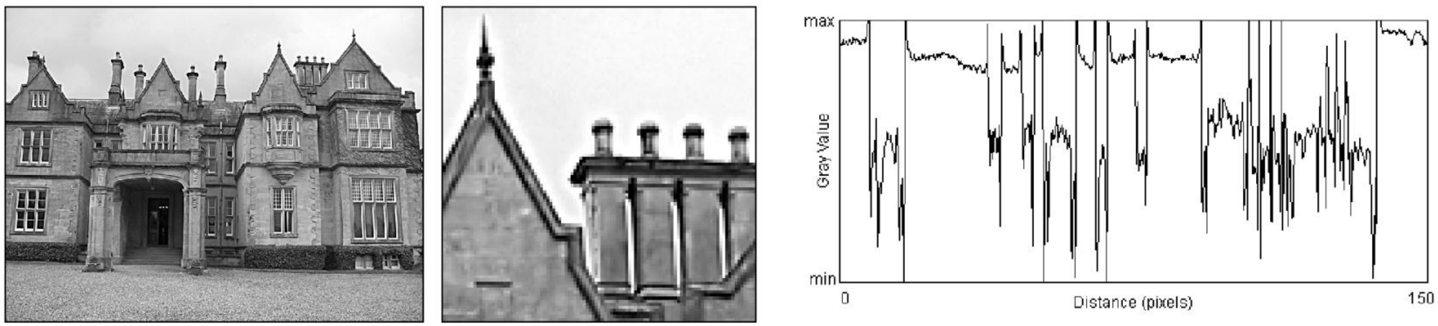
- Ostri tudi šum v homogenih regijah
- Razširitev: ostrenje samo kjer je lokalni kontrast dovolj velik:

$$\check{I}(u,v) \leftarrow \begin{cases} I(u,v) + a \cdot M(u,v) & \text{for } |\nabla I|(u,v) \geq t_c \\ I(u,v) & \text{otherwise.} \end{cases}$$

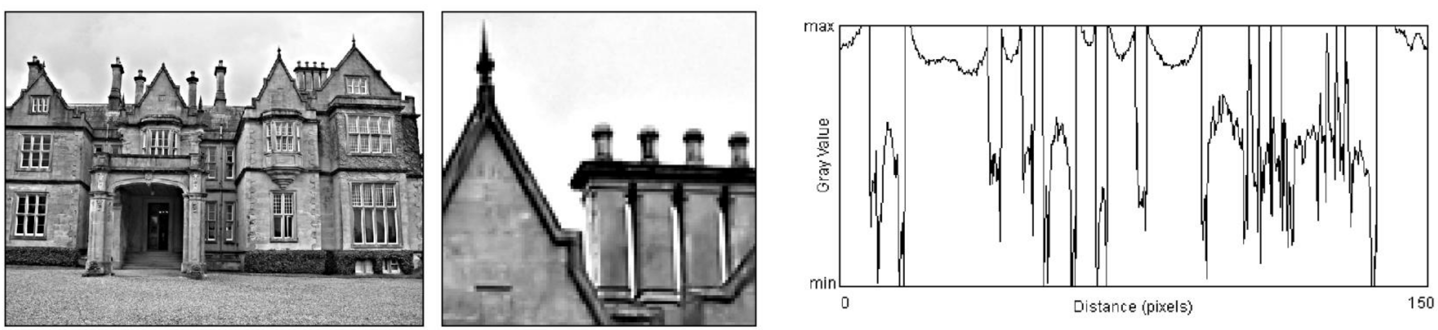
Učinek USM filtra



(a) Original (b) (c)



(d) $\sigma = 2.5$ (e) (f)



(g) $\sigma = 10.0$ (h) (i)

Laplaceov proti USM filtru

- Ostrenje z Laplaceovim filtrom je poseben primer USM filtra s filtrom za glajenje:

$$\tilde{H} = \tilde{H}^L = \frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$H^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 \end{bmatrix} - 5\begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= 5\left(\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) = 5\left(\tilde{H} - \delta\right)$$

$$\begin{aligned} \check{I}_L &\leftarrow I - w \cdot (H^L * I) = I - w \cdot \left(5(\tilde{H}^L - \delta) * I\right) \\ &= I - 5w \cdot (\tilde{H}^L * I - I) = I + 5w \cdot (I - \tilde{H}^L * I) \\ &= I + 5w \cdot M^L \end{aligned}$$

Detekcija kotov

- Koti so zelo pomembni, izstopajoči strukturni elementi na slikah
- So robustni:
  - Se ne pojavljajo naključno
  - So ponovljivi
  - Se jih da robustno zaznati
    - na različnih slikah
    - zajetimi pod različnimi koti
    - v različnih svetlobnih pogojih
    - na različnih razdaljah
- Detekcija kotov se zelo pogosto uporablja:
  - Za sledenje predmetom
  - Za ugotavljanje korespondence med stereo pari slik
  - Kot referenčne točke za natančne geometrične meritve
  - Za kalibracijo sistemov kamer za strojni vid
  - Za ugotavljanje ujemanja med slikami
  - Za iskanje točk kjer se aplicirajo bolj zahtevni operatorji

Zanimive točke

- Koti so zelo primerni za iskanje zanimivih točk (points of interest)
- Zahtevane lastnosti za detektor kotov:
  - Razlikovanje med pravimi in slučajnimi koti
  - Zanesljivost ob prisotnosti šuma
  - Natančno določanje lokacije kota
  - Učinkovita implementacija

- Kot je lokacija na sliki, kjer je gradient velik v več smereh hkrati
  - Večina metod za detekcijo kotov temelji na procesiranju prvih in drugih parcialnih odvodov

Harrisov detektor kotov

- Matrika lokalne strukture
  - Temelji na prvih parcialnih odvodih

$$I_x(u,v) = \frac{\partial I}{\partial x}(u,v) \quad \text{and} \quad I_y(u,v) = \frac{\partial I}{\partial y}(u,v)$$

$$A(u,v) = I_x^2(u,v)$$

$$B(u,v) = I_y^2(u,v)$$

$$C(u,v) = I_x(u,v) \cdot I_y(u,v)$$

$$\boldsymbol{M} = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

Harrisov detektor kotov

- Glajenje elementov matrike lokalne strukture

$$\bar{M} = \begin{pmatrix} A * H^{G,\sigma} & C * H^{G,\sigma} \\ C * H^{G,\sigma} & B * H^{G,\sigma} \end{pmatrix} = \begin{pmatrix} \bar{A} & \bar{C} \\ \bar{C} & \bar{B} \end{pmatrix}$$

- Diagonalizacija

$$\bar{M}' = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$\begin{aligned} \lambda_{1,2} &= \frac{\mathrm{trace}(\bar{M})}{2} \pm \sqrt{\left(\frac{\mathrm{trace}(\bar{M})}{2}\right)^2 - \det(\bar{M})} \\ &= \frac{1}{2}\left(\bar{A} + \bar{B} \pm \sqrt{\bar{A}^2 - 2\bar{A}\bar{B} + \bar{B}^2 + 4\bar{C}^2}\right) \end{aligned}$$

- Lastne vrednosti vsebujejo zelo pomembno informacijo o lokalni strukturi slike
  - Če sta obe dovolj veliki (sta si podobni) je verjetnost za kot velika

Harrisov detektor kotov

- Funkcija odziva na kote (Corner Response Function – CRF)
  - Temelji na razliki med lastnima vrednostima

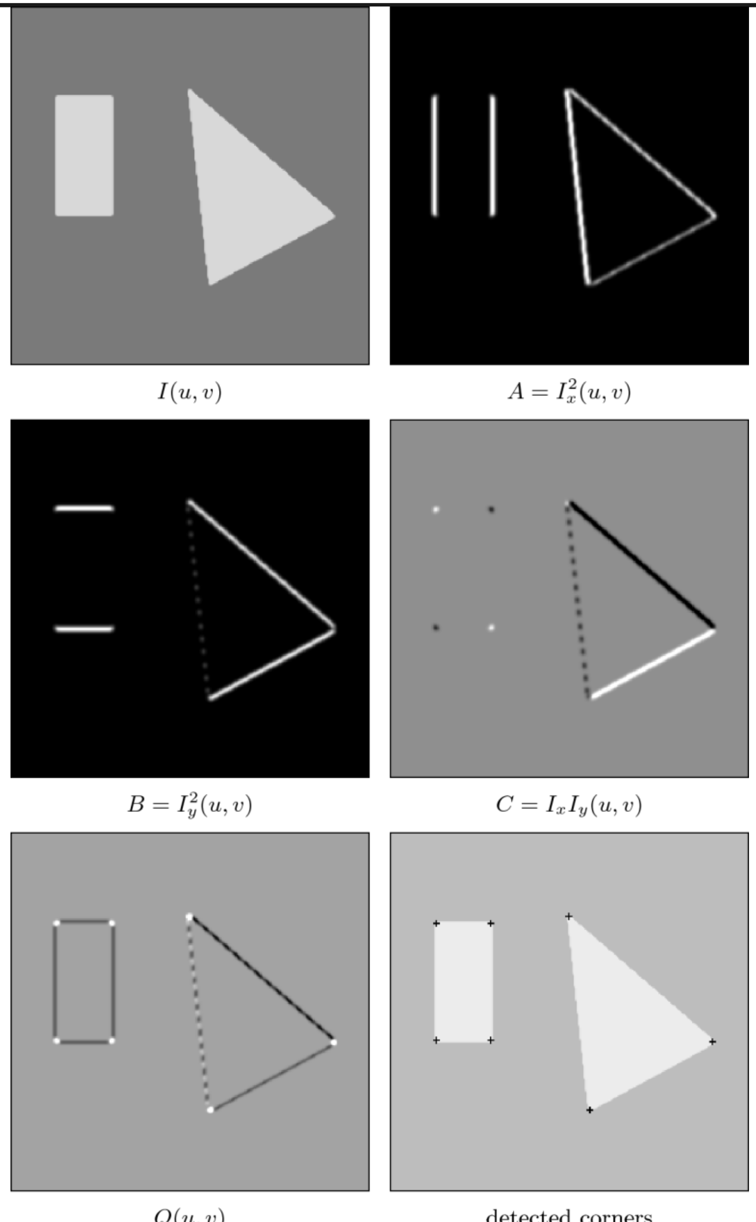
$$\lambda_1 - \lambda_2 = 2 \cdot \sqrt{\frac{1}{4} \cdot \left(\mathrm{trace}(\bar{M})\right)^2 - \det(\bar{M})}$$

  - Mera za jakost kota:

$$\begin{aligned} Q(u,v) &= \det(\bar{M}) - \alpha \cdot \left(\mathrm{trace}(\bar{M})\right)^2 \\ &= (\bar{A}\bar{B} - \bar{C}^2) - \alpha \cdot (\bar{A} + \bar{B})^2 \end{aligned}$$

  - Parameter *α* določa občutljivost detektorja
    - Večji kot je, manj občutljiv je detektor, manj kotov je detektiranih
- Določanje kotov
  - Izberejo se točke z $Q(u,v) > t_H$
  - Dobimo seznam točk $Corners = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots \boldsymbol{c}_N]$
  - Seznam sortiramo po močeh kotov $Q(u,v)$ ohranimo samo najmočnejše v vsaki oolici

Ilustrativni primer



Algoritem

1: HarrisCorners(I)

Returns a list of the strongest corners found in the image I.

2: Step 1—compute the corner response function:

3: Prefilter (smooth) the original image: $I' \leftarrow I * H_p$

4: Compute the horizontal and vertical image derivatives:

$I_x \leftarrow I' * H_{dx}$

$I_y \leftarrow I' * H_{dy}$

5: Compute the local structure matrix $M(u,v) = \left(\begin{smallmatrix} A & C \\ C & B \end{smallmatrix}\right)$:

$A(u,v) \leftarrow I_x^2(u,v)$

$B(u,v) \leftarrow I_y^2(u,v)$

$C(u,v) \leftarrow I_x(u,v) \cdot I_y(u,v)$

6: Blur each component of the structure matrix: $\bar{M} = \left(\begin{smallmatrix} \bar{A} & \bar{C} \\ \bar{C} & \bar{B} \end{smallmatrix}\right)$:

$\bar{A} \leftarrow A * H_b$

$\bar{B} \leftarrow B * H_b$

$\bar{C} \leftarrow C * H_b$

7: Compute the corner response function:

$Q \leftarrow (\bar{A} \cdot \bar{B} - \bar{C}^2) - \alpha \cdot (\bar{A} + \bar{B})^2$

8: Step 2—collect corner points:

9: Create an empty list:

Corners $\leftarrow [\,]$

10: **for** all image coordinates (u,v) **do**

11: **if** $Q(u,v) > t_H$ **and** IsLocalMax(Q,u,v) **then**

12: Create a new corner:

$\boldsymbol{c}_i \leftarrow \langle u_i, v_i, q_i \rangle = \langle u, v, Q(u,v) \rangle$

13: Add $\boldsymbol{c}_i$ to *Corners*

14: **Sort** *Corners* by q_i in *descending* order (strongest corners first)

15: *GoodCorners* $\leftarrow$ CleanUpNeighbors(*Corners*)

16: **return** *GoodCorners*.

Prefilter (line 3): Smoothing with a small xy-separable filter $H_p = H_{px} * H_{py}$, where

$$H_{px} = \frac{1}{9}\begin{bmatrix} 2 & \mathbf{5} & 2 \end{bmatrix} \quad \text{and} \quad H_{py} = H_{px}^T = \frac{1}{9}\begin{bmatrix} 2 \\ \mathbf{5} \\ 2 \end{bmatrix}.$$

Gradient filter (line 4): Computing the first partial derivative in the x and y directions with

$$H_{dx} = \begin{bmatrix} -0.453014 & \mathbf{0} & 0.453014 \end{bmatrix} \quad \text{and} \quad H_{dy} = H_{dx}^T = \begin{bmatrix} -0.453014 \\ \mathbf{0} \\ 0.453014 \end{bmatrix}.$$

Blurfilter (line 6): Smoothing the individual components of the structure matrix M with separable Gaussian filters $H_b = H_{bx} * H_{by}$

$$H_{bx} = \frac{1}{64}\begin{bmatrix} 1 & 6 & 15 & \mathbf{20} & 15 & 6 & 1 \end{bmatrix}, \qquad H_{by} = H_{bx}^T = \frac{1}{64}\begin{bmatrix} 1 \\ 6 \\ 15 \\ \mathbf{20} \\ 15 \\ 6 \\ 1 \end{bmatrix}$$

Steering parameter (line 7): $\alpha = 0.04$ to 0.06 (default 0.05)

Response threshold (line 13): $t_H = 10{,}000$ to $1{,}000{,}000$ (default 25,000)

Algoritem

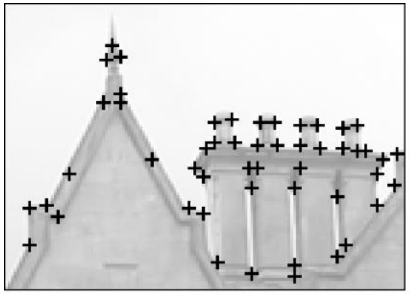
17: $\textsc{IsLocalMax}(Q, u, v)$ ▷ determine if $Q(u, v)$ is a local maximum
18: Let $q_c \leftarrow Q(u, v)$ (center pixel)
19: Let $\mathcal{N} \leftarrow \mathit{Neighbors}(Q, u, v)$ ▷ values of all neighboring pixels
20: **if** $q_c \geq q_i$ for all $q_i \in \mathcal{N}$ **then**
21: **return** *true*
22: **else**
23: **return** *false*.

24: $\textsc{CleanUpNeighbors}(\mathit{Corners})$ ▷ *Corners* is sorted by descending q
25: Create an empty list:
$\mathit{GoodCorners} \leftarrow [\,]$
26: **while** *Corners* is not empty **do**
27: $\boldsymbol{c}_i \leftarrow \textsc{RemoveFirst}(\mathit{Corners})$
28: Add $\boldsymbol{c}_i$ to *GoodCorners*
29: **for** all $\boldsymbol{c}_j$ in *Corners* **do**
30: **if** $\mathit{Dist}(\boldsymbol{c}_i, \boldsymbol{c}_j) < d_{\min}$ **then**
31: Delete $\boldsymbol{c}_j$ from *Corners*
32: **return** *GoodCorners*.

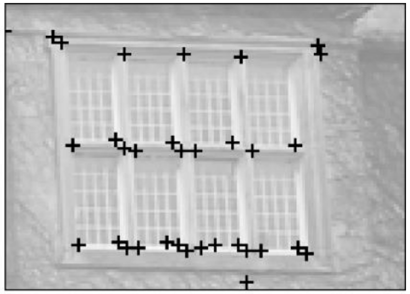
Neighborhood radius (line 31): $d_{\min} = 10$ pixels

Primer

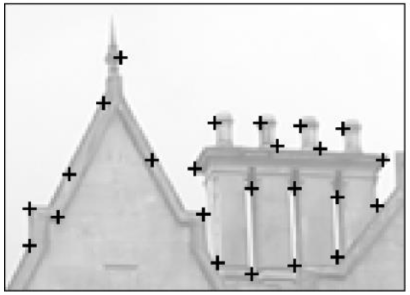
(a)



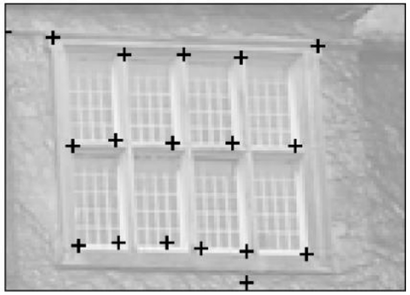
(b)



(c)



(d)



(e)