# Robotika in računalniško zaznavanje (RRZ)

# Regije

Danijel Skočaj

Univerza v Ljubljani

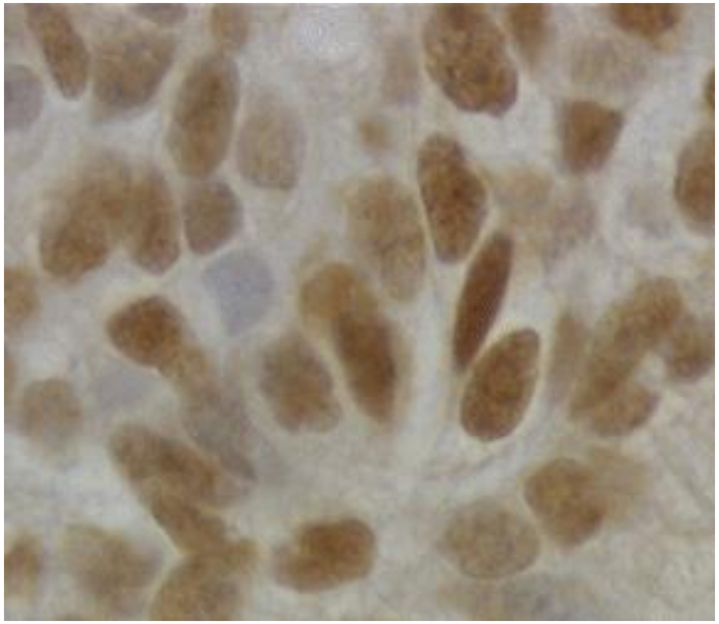
Fakulteta za računalništvo in informatiko

Literatura: W. Burger, M. J. Burge (2008).
Digital Image Processing, poglavja 10, 11

v7.0

# Regije in filtriranje regij

- Kako bi izvedli algoritem, ki bi avtomatsko preštel število podolgovatih celic v sliki?

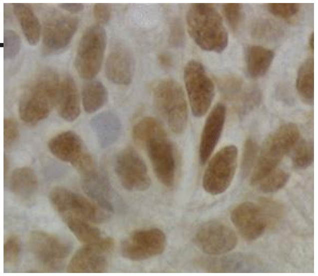


$$n=?$$

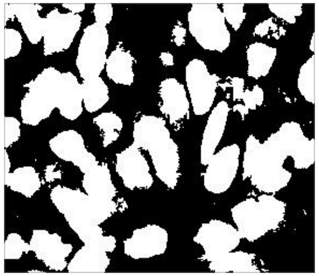
# Zaporedje postopkov

- Spremeni sliko v binarno
  - Upragovljenje
- Čiščenje binarne slike
  - Morfološke operacije
- Izločanje posameznih predmetov
  - Labeliranje
- Opis vsake regije posebej in klasifikacija po obliki

izvorna slika

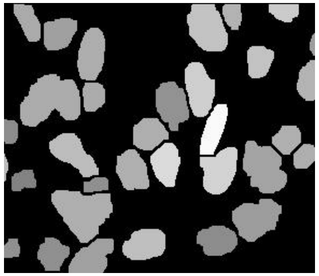


Upragovljena slika



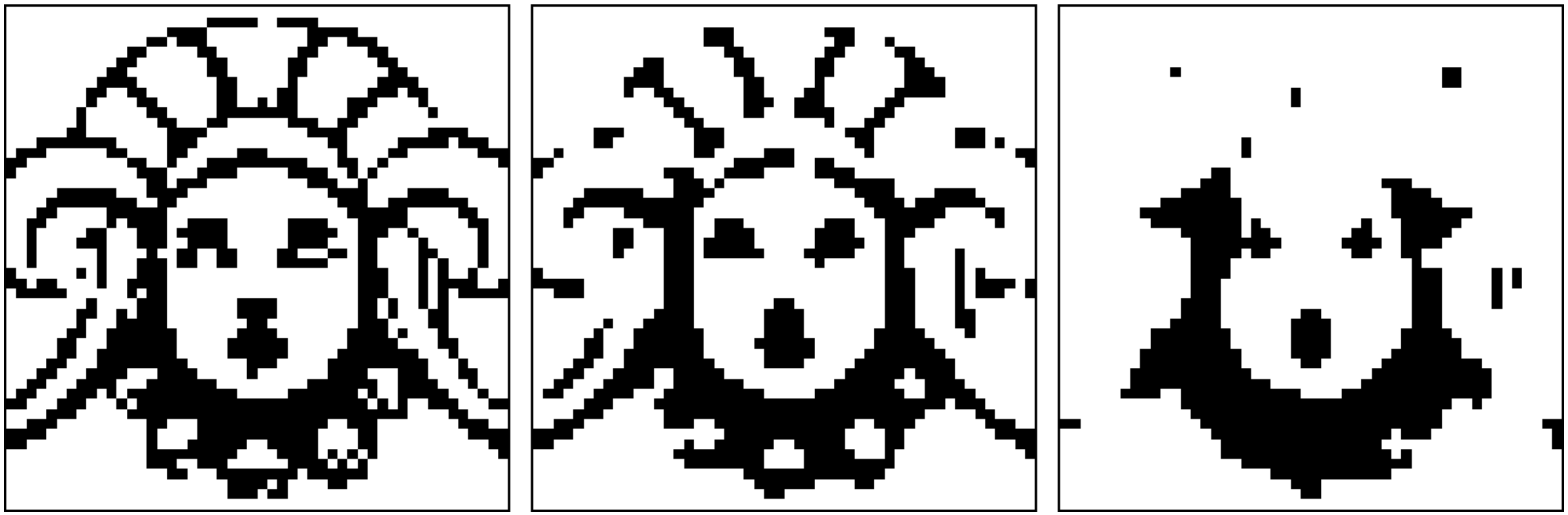
Očiščena slika

Labelirana slika



# Morfološki filtri

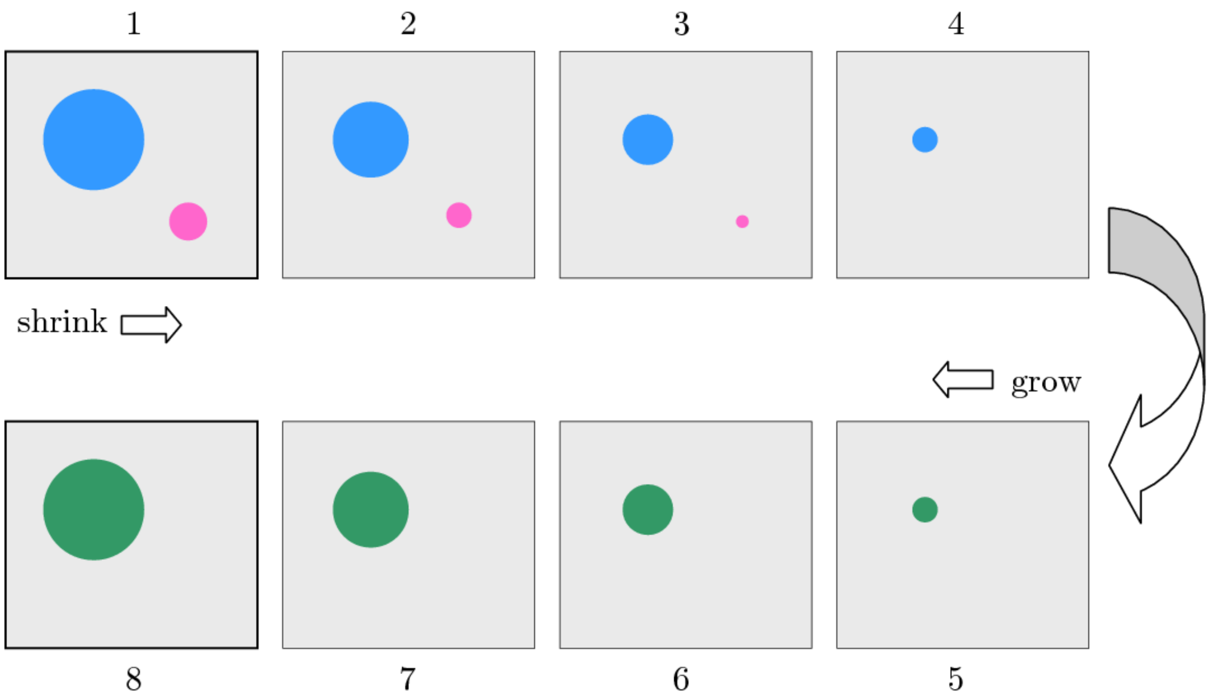
- Linearni filtri ne spreminjajo topologije slike
- Medianin filter deloma spremeni strukturo slike:



- Morfološki filtri so namenjeni ravno spreminjanju lokalne strukture slike
  - Odstranjevanje majhnih elementov slike
  - Polnjenje lukenj
  - Iskanje obrisov

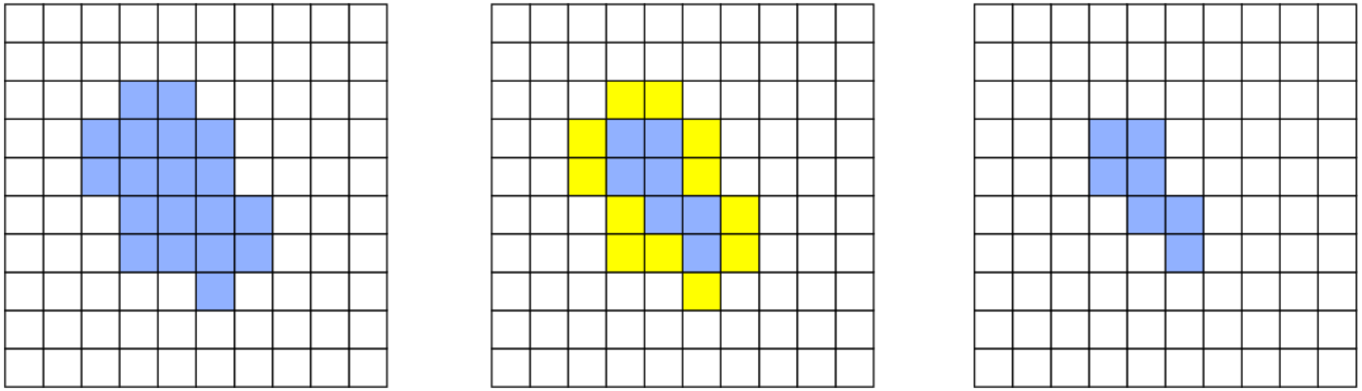
# Krčenje in rast

- Odstranjevanje majhnih elementov na sliki
- Algoritem:
  1. Vse strukture skrči, tako da se odstranijo zunanji deli struktur
     - Na ta način se izgubijo majhne strukture
  2. Preostale struture naj spet zrastejo – se dodajajo zunanje plasti
     - Doseżejo približno enako velikost in obliko kot na začetku

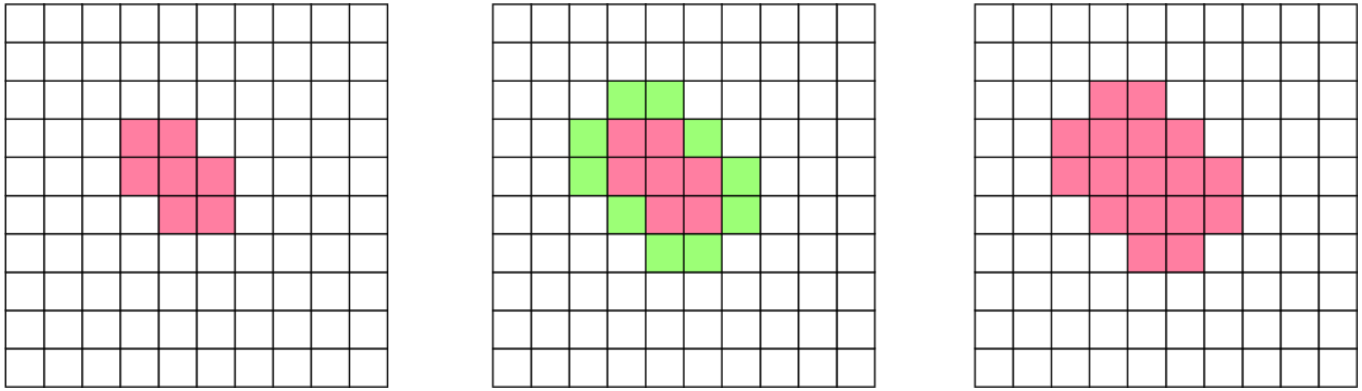


# Krčenje in rast

- Krčenje
  - Odstrani se zunanja plast slikovnih elementov

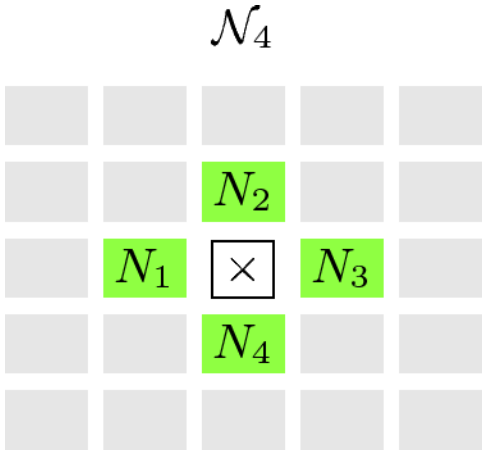


- Rast
  - doda se zunanja plast slikovnih elementov



# Sosednost slikovnih elementov

- 4-sosednost




- 8-sosednost

# Osnovne morfološke operacije

- Krčenje in Rast (Shrinking in Growing)
- Erozija in Širitev (Erosion, Dilation)
  - Bolj splošne
  - Definirane s strukturnim elementom

- Strukturni element
  - Definira lastnosti morfološkega filtra
  - Matrika binarnih vrednosti

$$H(i,j) \in \{0,1\}$$

# Množice točk

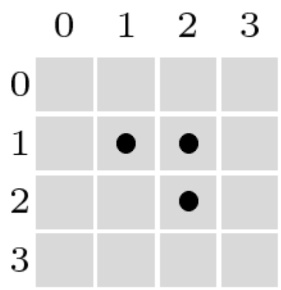
- Morfološke operacije ponavadi izvajamo na binarnih slikah

$$I(u,v) \in \{0,1\}$$

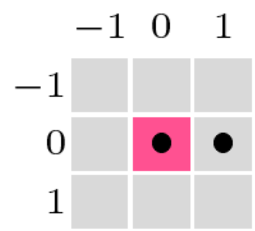
- Binarne slike in strukturni element lahko predstavimo z množico točk (parov koordinat $\boldsymbol{p} = (u,v)$ ):

$$\mathcal{Q}_I = \{\boldsymbol{p} \mid I(\boldsymbol{p}) = 1\}$$

$I$




$H$




$$I \equiv \mathcal{Q}_I = \{(1,1),(2,1),(2,2)\}$$

$$H \equiv \mathcal{Q}_H = \{(0,0),(1,0)\}$$

# Osnovne binarne operacije

- Invertiranje - komplementarna množica

$$\mathcal{Q}_{\bar{I}} = \bar{\mathcal{Q}}_I = \{\boldsymbol{p} \in \mathbb{Z}^2 \mid \boldsymbol{p} \notin \mathcal{Q}_I\}$$

- ALI – unija

$$\mathcal{Q}_{I_1 \vee I_2} = \mathcal{Q}_{I_1} \cup \mathcal{Q}_{I_2}$$

- Translacija – prištevanje

$$I_{\boldsymbol{d}} \equiv \left\{(\boldsymbol{p} + \boldsymbol{d}) \mid \boldsymbol{p} \in I\right\}$$

- Zrcaljenje - negacija

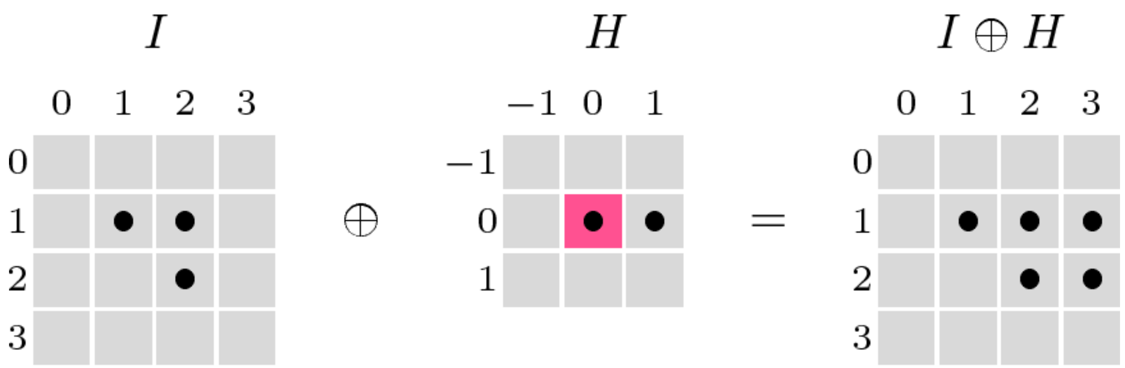
$$H^* \equiv \{-\boldsymbol{p} \mid \boldsymbol{p} \in H\}$$

# Širitev

- Iz Rasti - strukturni element se razmnoži na vsakem slikovnem elementu ospredja

$$I \oplus H \equiv \left\{ (\boldsymbol{p} + \boldsymbol{q}) \mid \text{for some } \boldsymbol{p} \in I \text{ and } \boldsymbol{q} \in H \right\}$$

$$I \oplus H \equiv \bigcup_{\boldsymbol{p} \in I} H_{\boldsymbol{p}} = \bigcup_{\boldsymbol{q} \in H} I_{\boldsymbol{q}}$$




$$I \equiv \{(1,1),(2,1),(2,2)\}, \;\; H \equiv \{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{0})\}$$

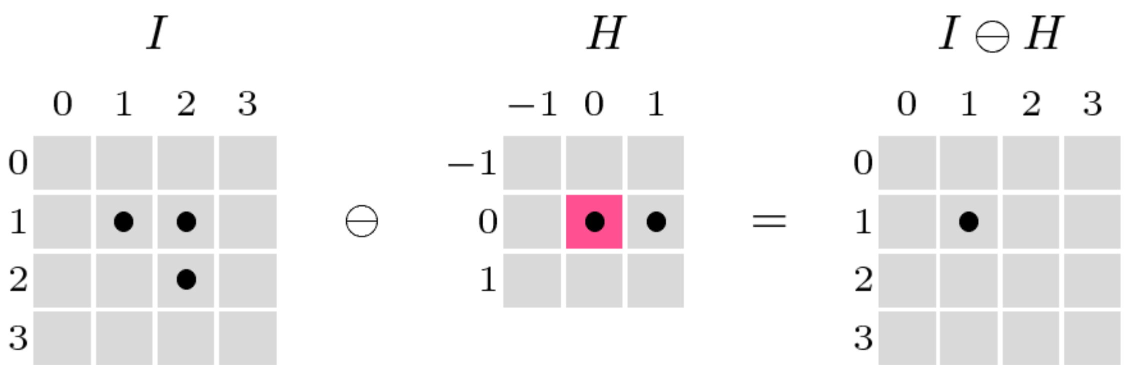
$$\begin{aligned} I \oplus H \;\; \equiv \;\; \{ \; & (1,1)+(\mathbf{0},\mathbf{0}), \; (1,1)+(\mathbf{1},\mathbf{0}), \\ & (2,1)+(\mathbf{0},\mathbf{0}), \; (2,1)+(\mathbf{1},\mathbf{0}), \\ & (2,2)+(\mathbf{0},\mathbf{0}), \; (2,2)+(\mathbf{1},\mathbf{0}) \; \} \end{aligned}$$

# Erozija

- Iz Krčenja - kvazi-inverz Širitve

$$I \ominus H \equiv \{\boldsymbol{p} \in \mathbb{Z}^2 \mid (\boldsymbol{p} + \boldsymbol{q}) \in I, \text{ for every } \boldsymbol{q} \in H\}$$

$$I \ominus H \equiv \{\boldsymbol{p} \in \mathbb{Z}^2 \mid H_{\boldsymbol{p}} \subseteq I\}$$




$$I \equiv \{(1,1),(2,1),(2,2)\}, \; H \equiv \{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{0})\}$$

$$I \ominus H \equiv \{\,(1,1)\,\} \text{ because}$$

$$(1,1) + (\mathbf{0},\mathbf{0}) = (1,1) \in I \quad \textbf{and} \quad (1,1) + (\mathbf{1},\mathbf{0}) = (2,1) \in I$$

# Lastnosti Širitve in Erozije

- Komutativnost

$$I \oplus H = H \oplus I$$

$$I \ominus H \neq H \ominus I$$

- Asociativnost

$$(I_1 \oplus I_2) \oplus I_3 = I_1 \oplus (I_2 \oplus I_3)$$

$$H_{\text{big}} = H_1 \oplus H_2 \oplus \ldots \oplus H_K$$

$$I \oplus H_{\text{big}} = (\ldots((I \oplus H_1) \oplus H_2) \oplus \ldots \oplus H_K)$$

$$(I_1 \ominus I_2) \ominus I_3 = I_1 \ominus (I_2 \oplus I_3)$$

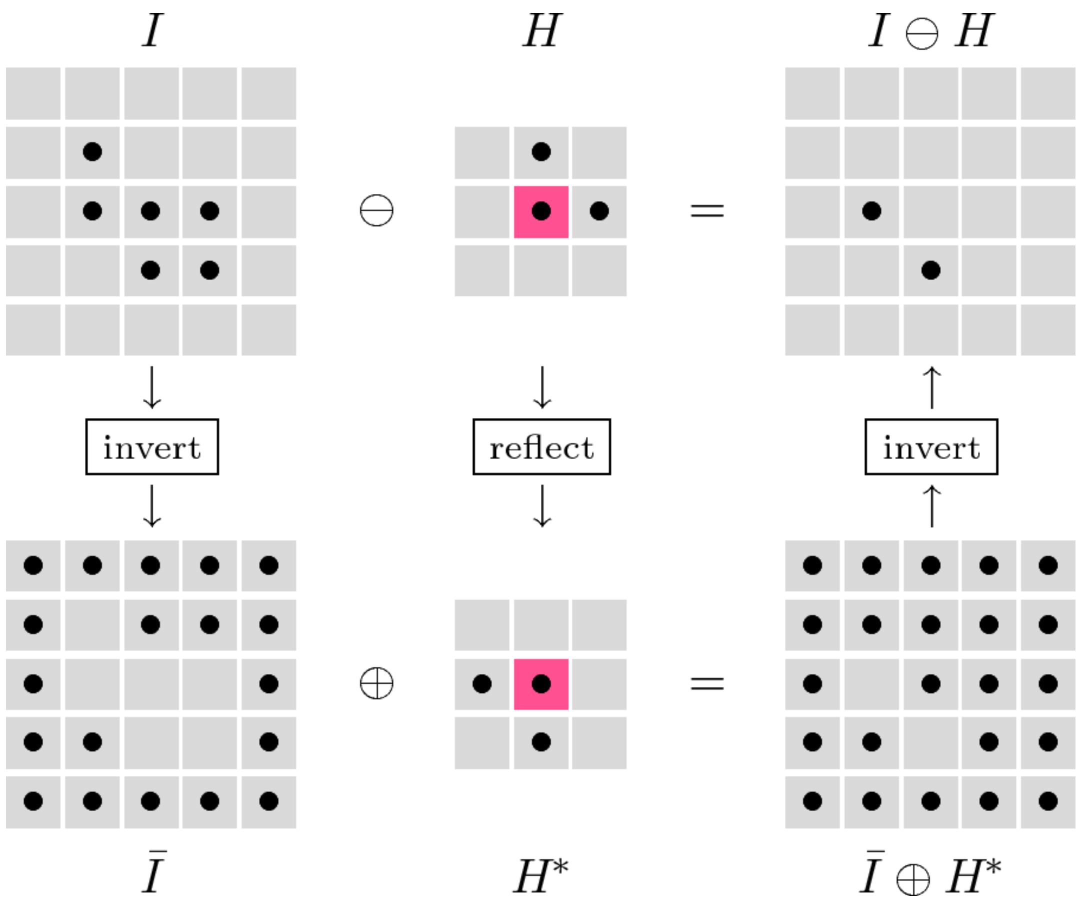
- Nevtralni element

$$I \oplus \delta = \delta \oplus I = I, \qquad \text{with } \delta \equiv \{(0,0)\}$$

# Dualnost Širitve in Erozije

- Širitev ospredja je enaka Eroziji ozadja in obratno

$$I \oplus H \equiv \overline{(\bar{I} \ominus H^*)} \qquad I \ominus H \equiv \overline{(\bar{I} \oplus H^*)}$$



# Algoritem

$\textsc{Dilate}\ (I, H)$

$I$: binary image of size $w \times h$
$H$: binary structuring element defined over region $\mathcal{R}_H$
Returns the dilated image $I' = I \oplus H$

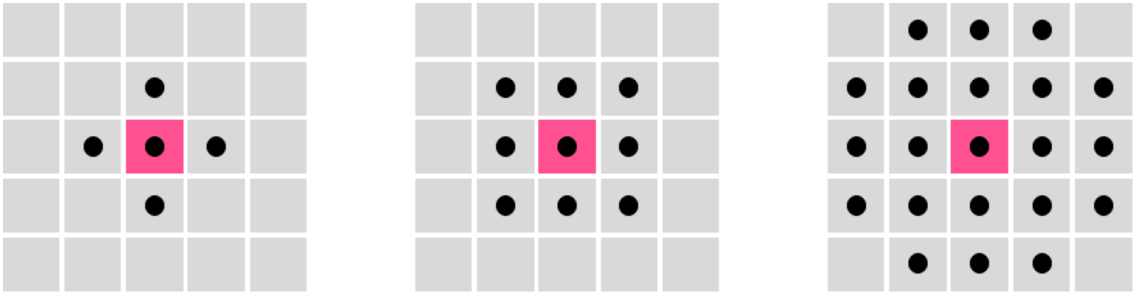
$I' \leftarrow$ new binary image of size $w \times h$
$I'(u,v) \leftarrow 0,$ for all $(u,v)$ $\quad \triangleright\ I' \leftarrow \varnothing$
**for all** $(i,j) \in \mathcal{R}_H$ **do** $\quad \triangleright\ (i,j) = \boldsymbol{q}$
  **if** $H(i,j) = 1$ **then** $\quad \triangleright\ \boldsymbol{q} \in H$
    $\textsc{Merge the shifted}\ I_{\boldsymbol{q}}\ \textsc{with}\ I'$: $\quad \triangleright\ I' \leftarrow I' \cup I_{\boldsymbol{q}}$
    **for** $u \leftarrow 0 \ldots (w{-}1)$ **do**
      **for** $v \leftarrow 0 \ldots (h{-}1)$ **do** $\quad \triangleright\ (u,v) = \boldsymbol{p}$
        **if** $I(u,v) = 1$ **then** $\quad \triangleright\ \boldsymbol{p} \in I$
          $I'(u{+}i, v{+}j) \leftarrow 1$ $\quad \triangleright\ I' \leftarrow I' \cup (\boldsymbol{p}{+}\boldsymbol{q})$
**return** $I'$.

$\textsc{Erode}\ (I, H)$

$\bar{I} \leftarrow \textsc{Invert}(I)$ $\quad \triangleright\ \bar{I} \leftarrow \neg I$
$H^* \leftarrow \textsc{Reflect}(H)$
**return** $\textsc{Invert}(\textsc{Dilate}(\bar{I}, H^*))$. $\quad \triangleright\ I \ominus H = \overline{(\bar{I} \oplus H^*)}$

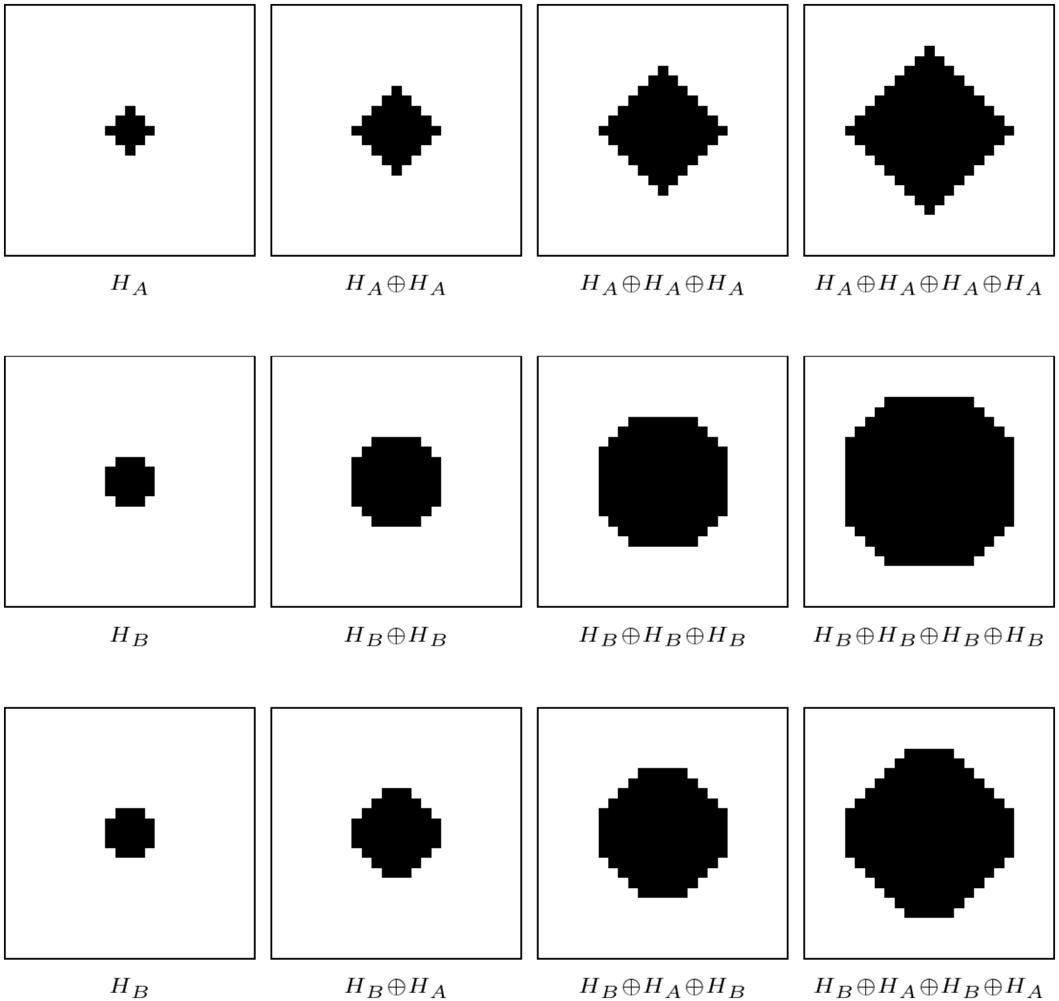
# Morfološki filtri

- Morfološki filter je točno definiran s
  - tipom operacije
  - strukturnim elementom
- Ponavadi se uporabljajo okrogli strukturni elementi (izotropičen filter)
  - Pri Širitvi strukturni element z radijem *r* doda *r* slikovnih elementov plasti okrog ospredja
  - Pri Eroziji strukturni element z radijem *r* zbriše *r* plasti slikovnih elementov z ospredja

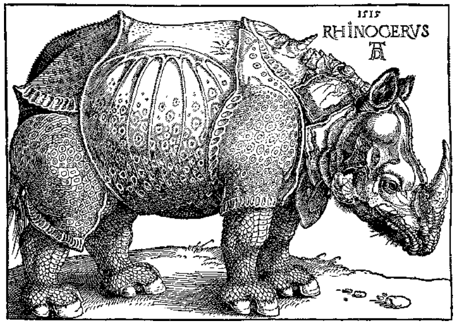


# Iterativnost filtrov

- Aproksimacija izotropičnih filtrov z ustreznim vrstnim redom manjših filtrov

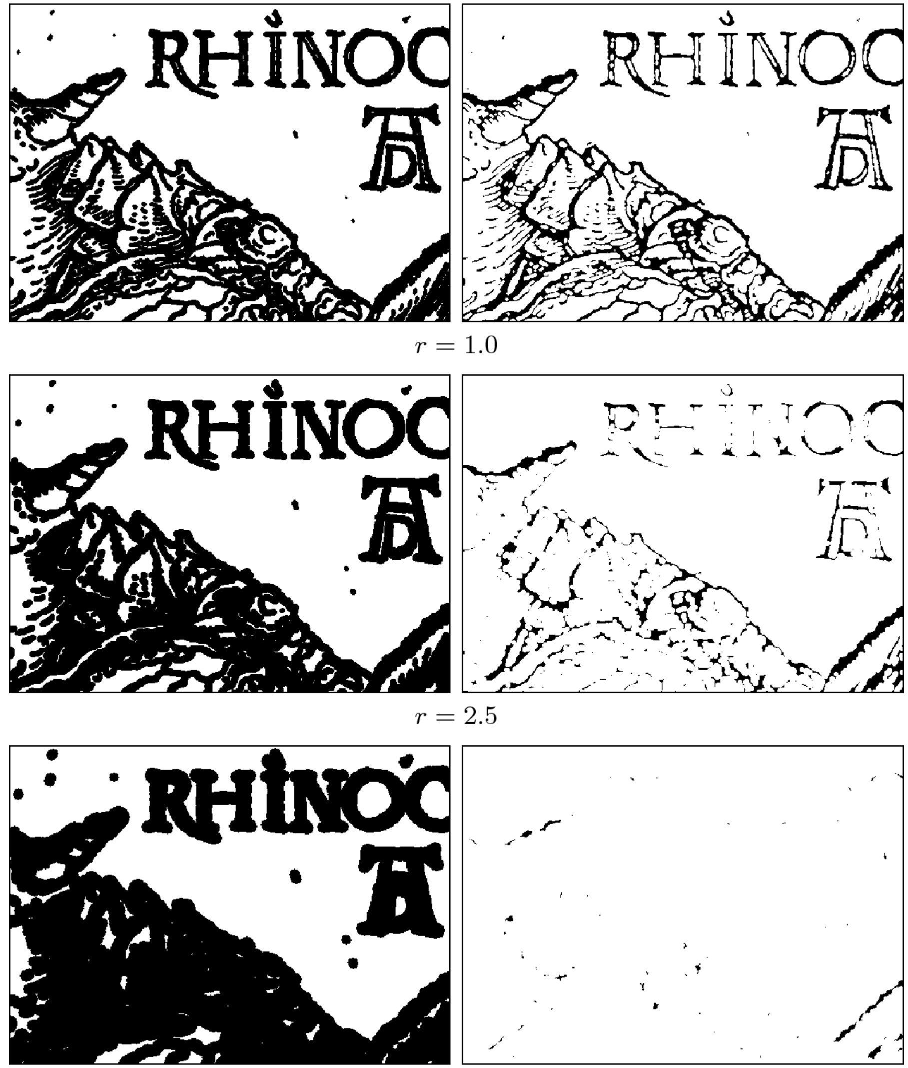


# Primer





Dilation Erosion

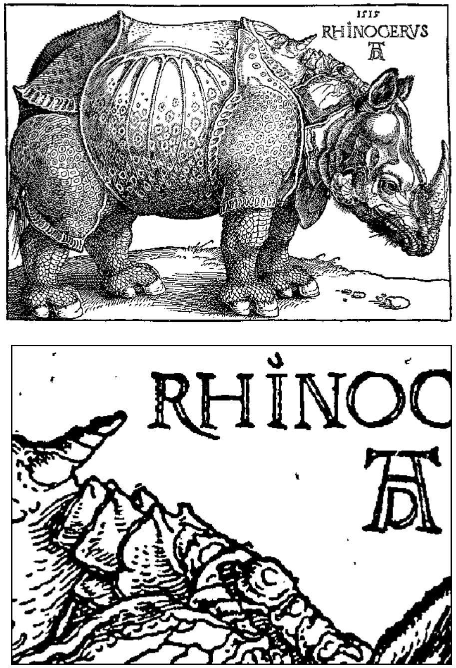


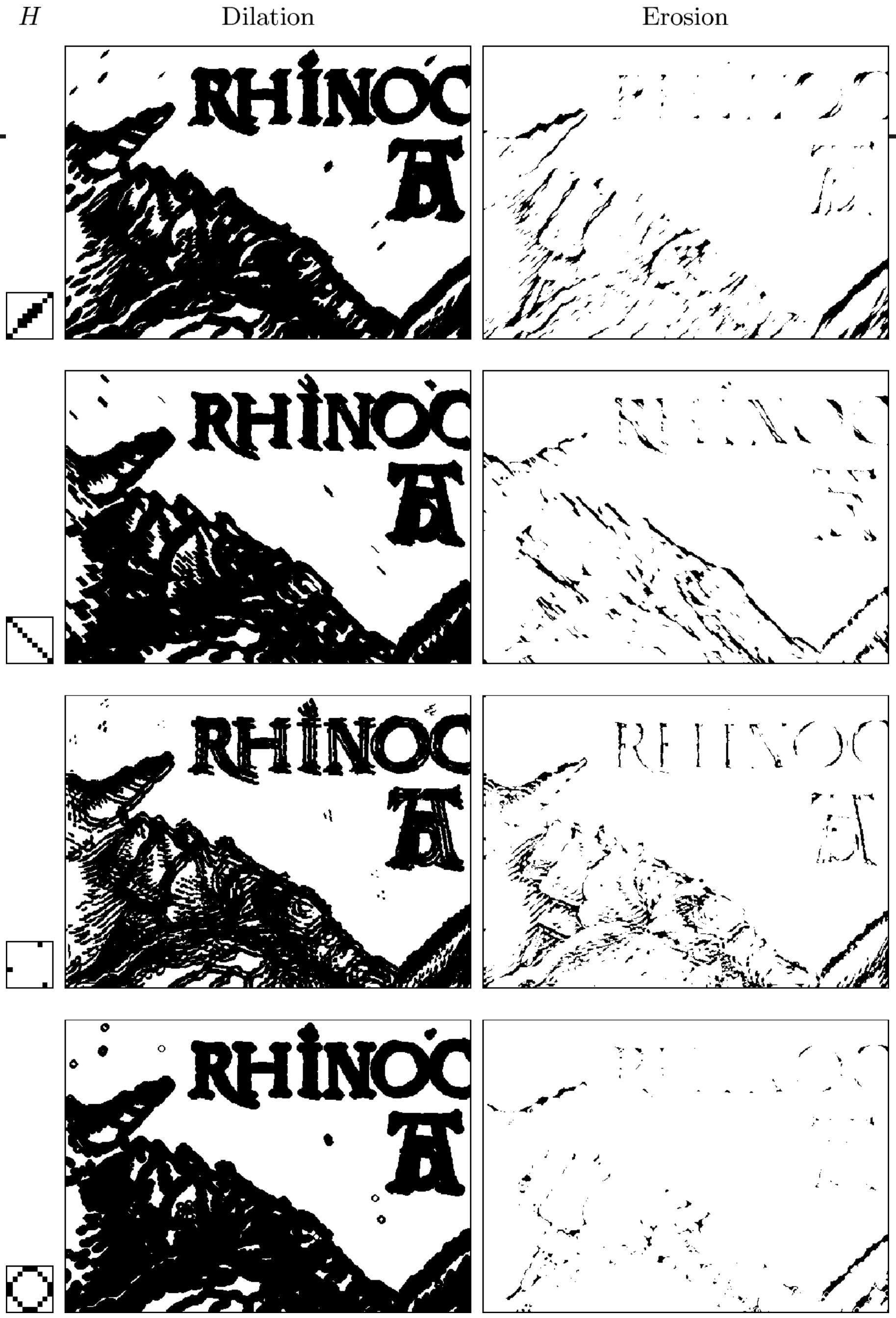

$r = 1.0$

$r = 2.5$

$r = 5.0$

# Primer



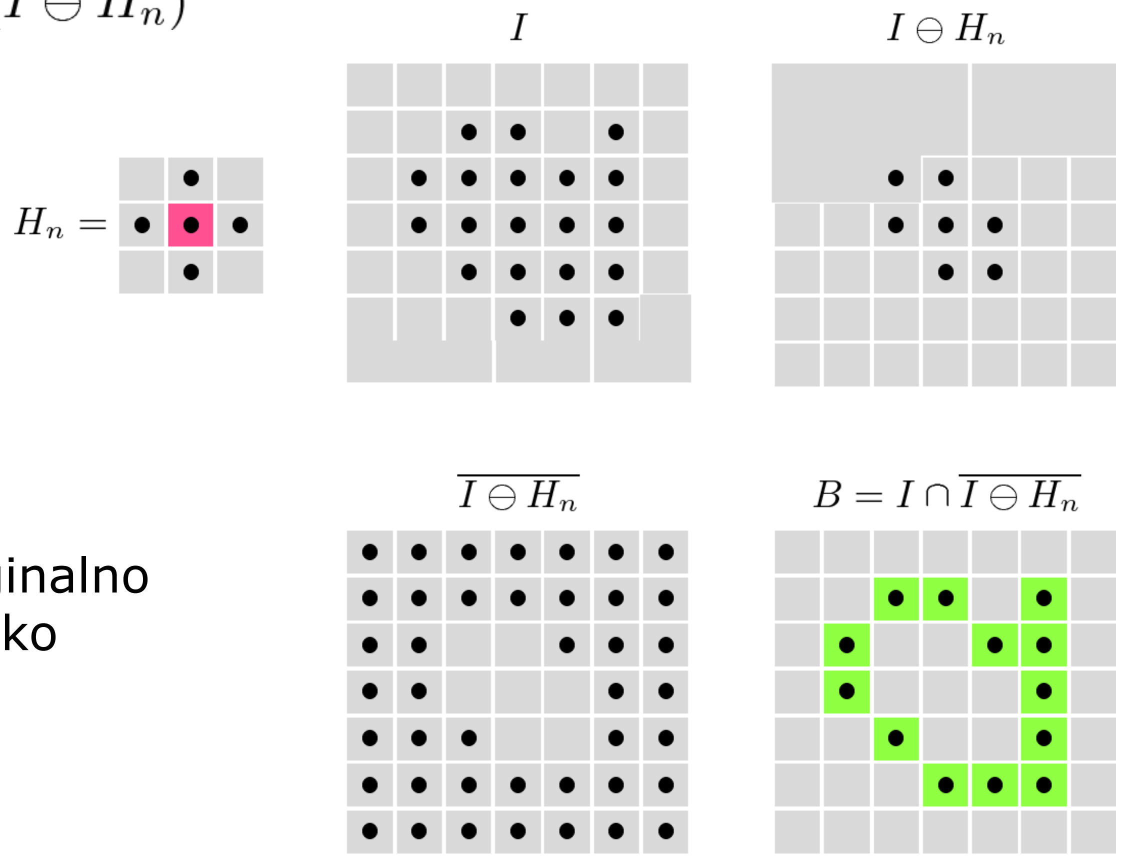


# Obrisi

- Določanje mejnih slikovnih elementov
  - Presek med originalno in invertirano „erodirano“ sliko

$$I' = I \ominus H_n$$

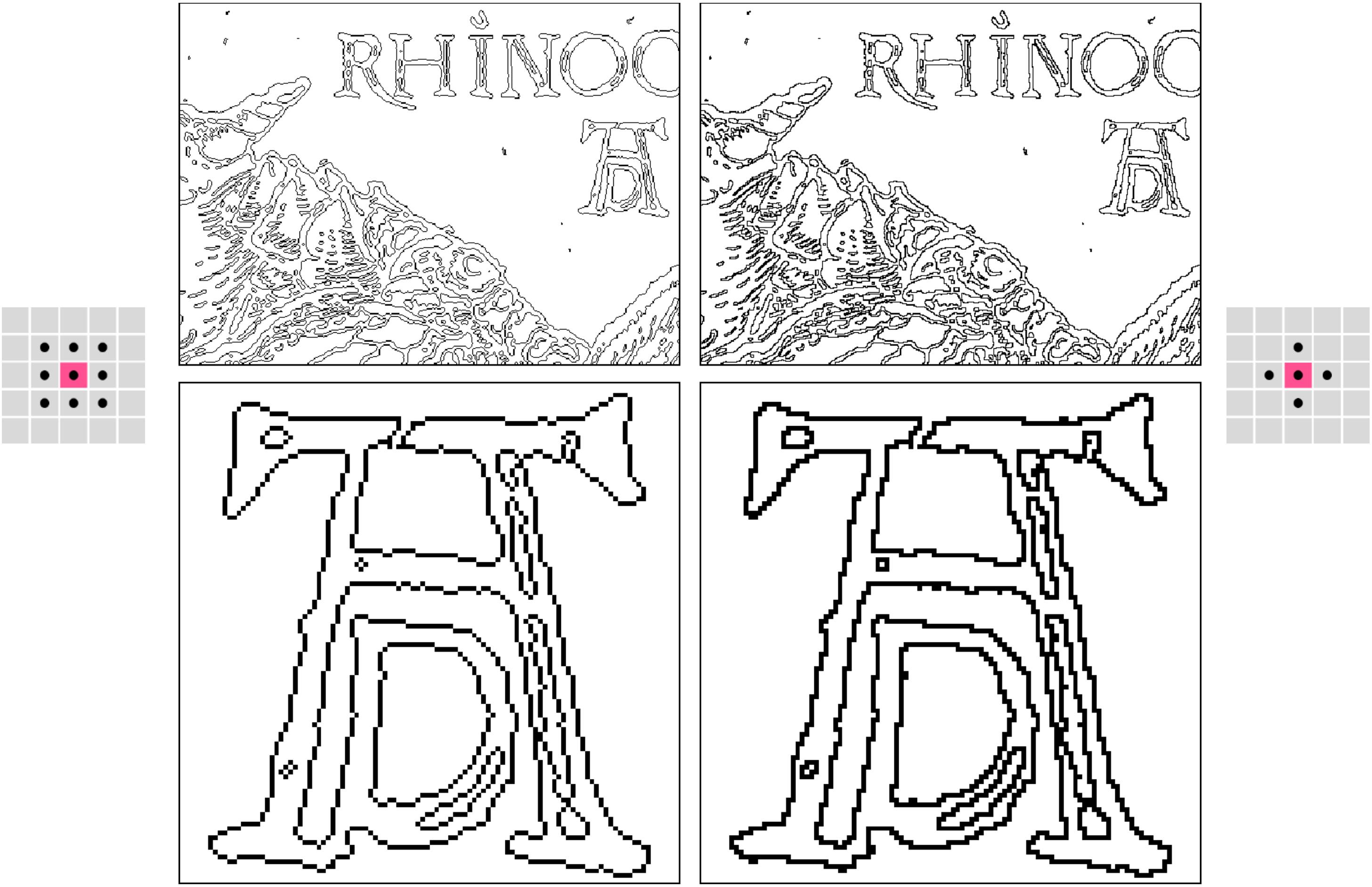
$$B = I \cap \overline{I'} = I \cap \overline{(I \ominus H_n)}$$




  - oz XOR med originalno in „erodirano“ sliko

# Primer

- 8-sosedni strukturni element -> 4-sosedni obris
- 4-sosedni strukturni element -> 8-sosedni obris



# Kompozitni operatorji

- Odprtje (opening):
  - Erozija, nato Širitev z istim strukturnim elementom
  - Za odstranjevanje majhnih elementov

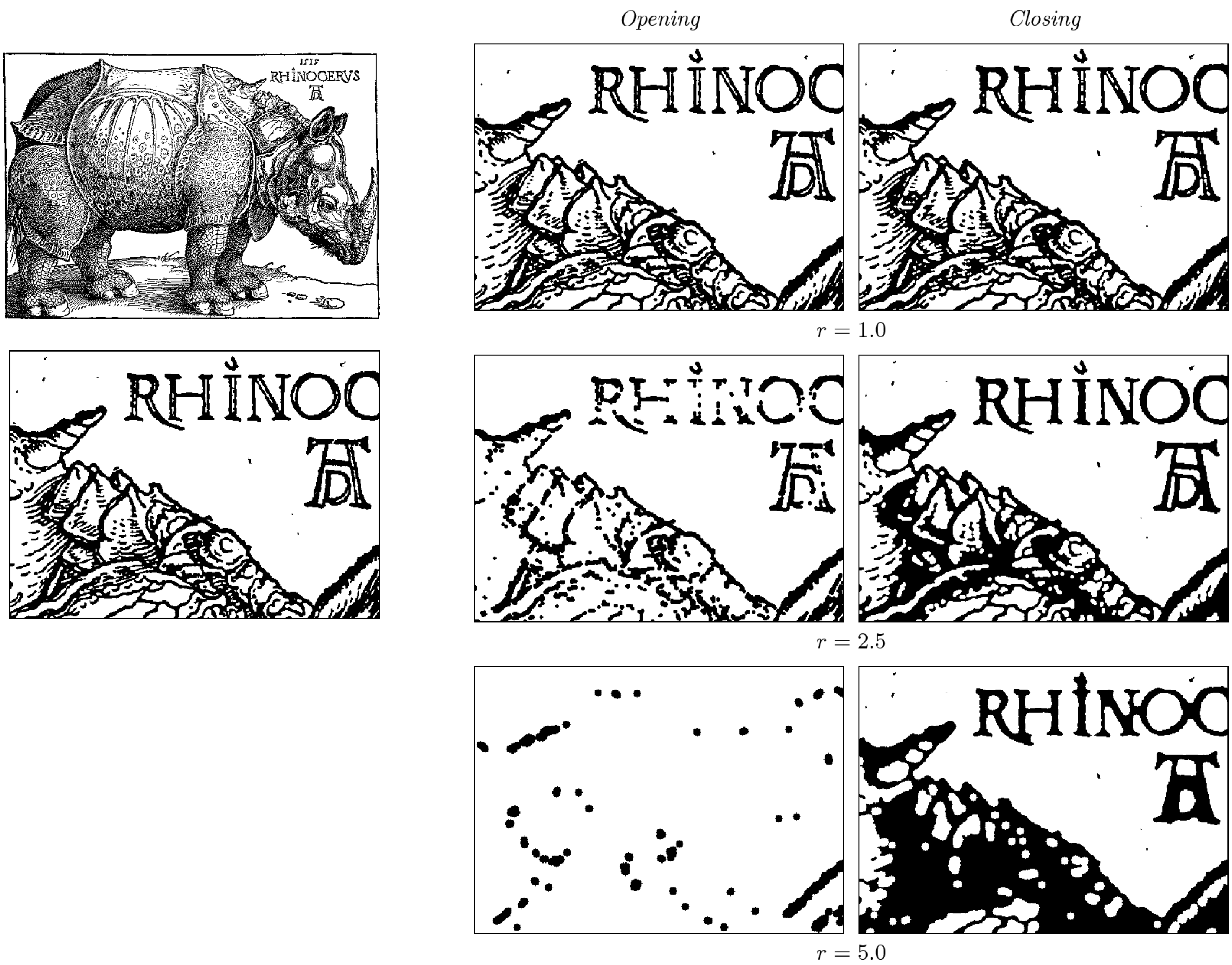
$$I \circ H = (I \ominus H) \oplus H$$

- Zaprtje (closing)
  - Širitev, nato erozija
  - Za polnjenje lukenj

$$I \bullet H = (I \oplus H) \ominus H$$

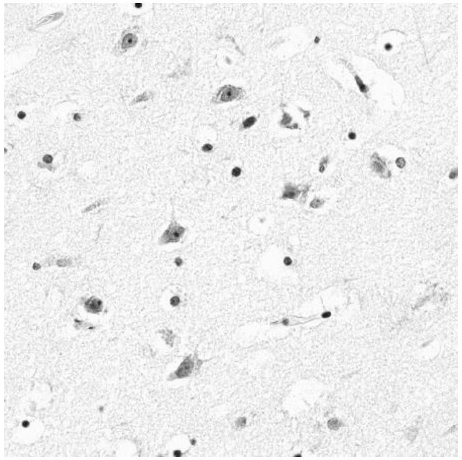
- Operaciji Odprtje in Zaprtje sta
  - Idempotentni:
    $$I \circ H = (I \circ H) \circ H = ((I \circ H) \circ H) \circ H = \ldots$$
    $$I \bullet H = (I \bullet H) \bullet H = ((I \bullet H) \bullet H) \bullet H = \ldots$$
  - Dualni: $I \circ H = \overline{(\bar{I} \bullet H)} \quad \text{and} \quad I \bullet H = \overline{(\bar{I} \circ H)}$

# Primer Odprtja in Zaprtja

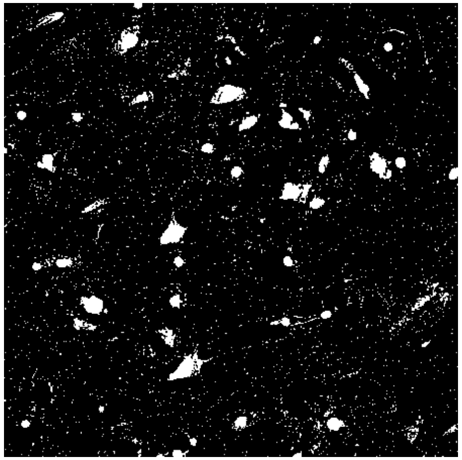


# Učinek odpiranja

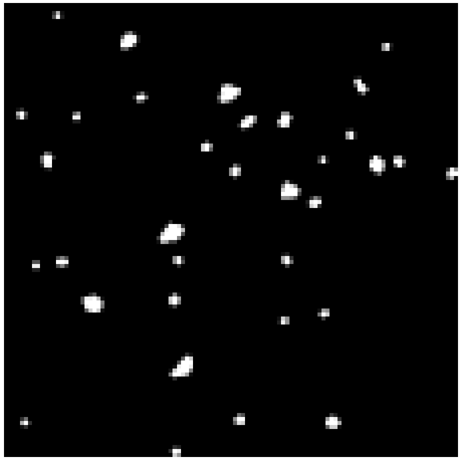
- Lahko filtriramo strukture s primerno izbiro velikosti strukturnega elementa



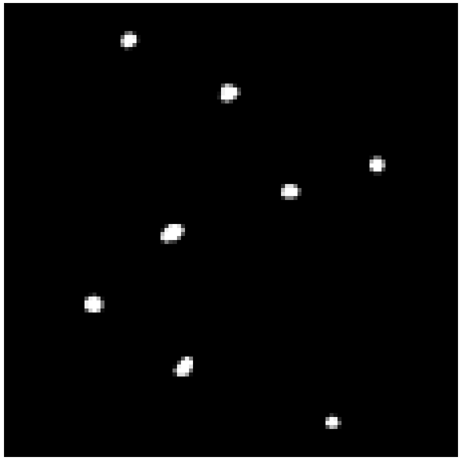
Originalna slika



Upragovljena slika



Odpiranje z majhnim strukturnim elementom

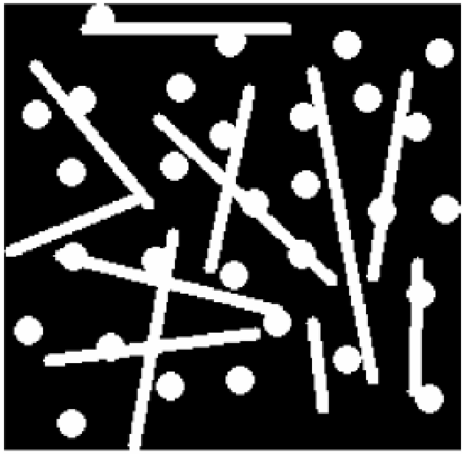


Odpiranje z velikim strukturnim elementom

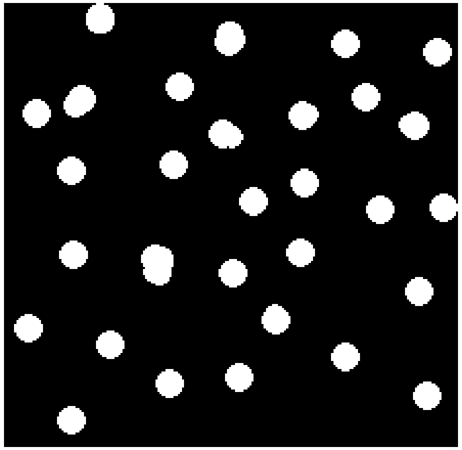
Vir slik: http://homepages.inf.ed.ac.uk/rbf/HIPR2/

# Učinek odpiranja

- Izbiramo strukture v sliki s pomočjo izbire oblike strukturnega elementa...



Vhodna slika



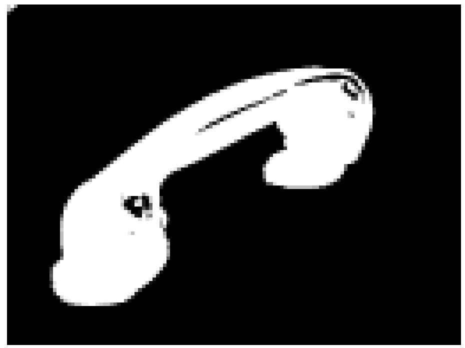
Odpiranje z okroglim strukturnim elementom

Vir slik: http://homepages.inf.ed.ac.uk/rbf/HIPR2/

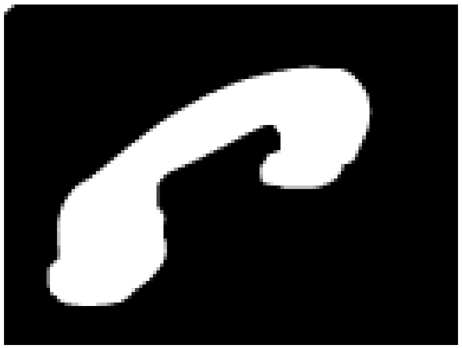
# Učinki zapiranja

- Zapolni luknje v upragovani sliki (npr., zaradi odbleskov)

Originalna slika

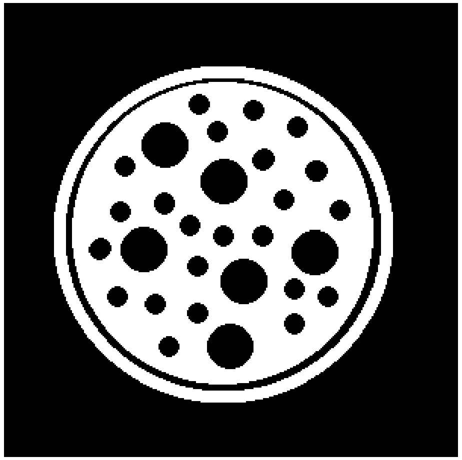


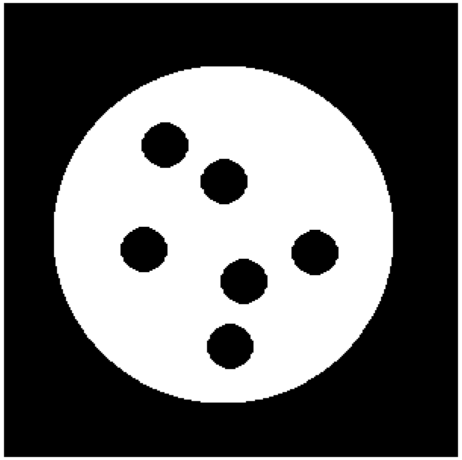
Upragovljena slika



Zapiranje z okroglim strukturnim elementom

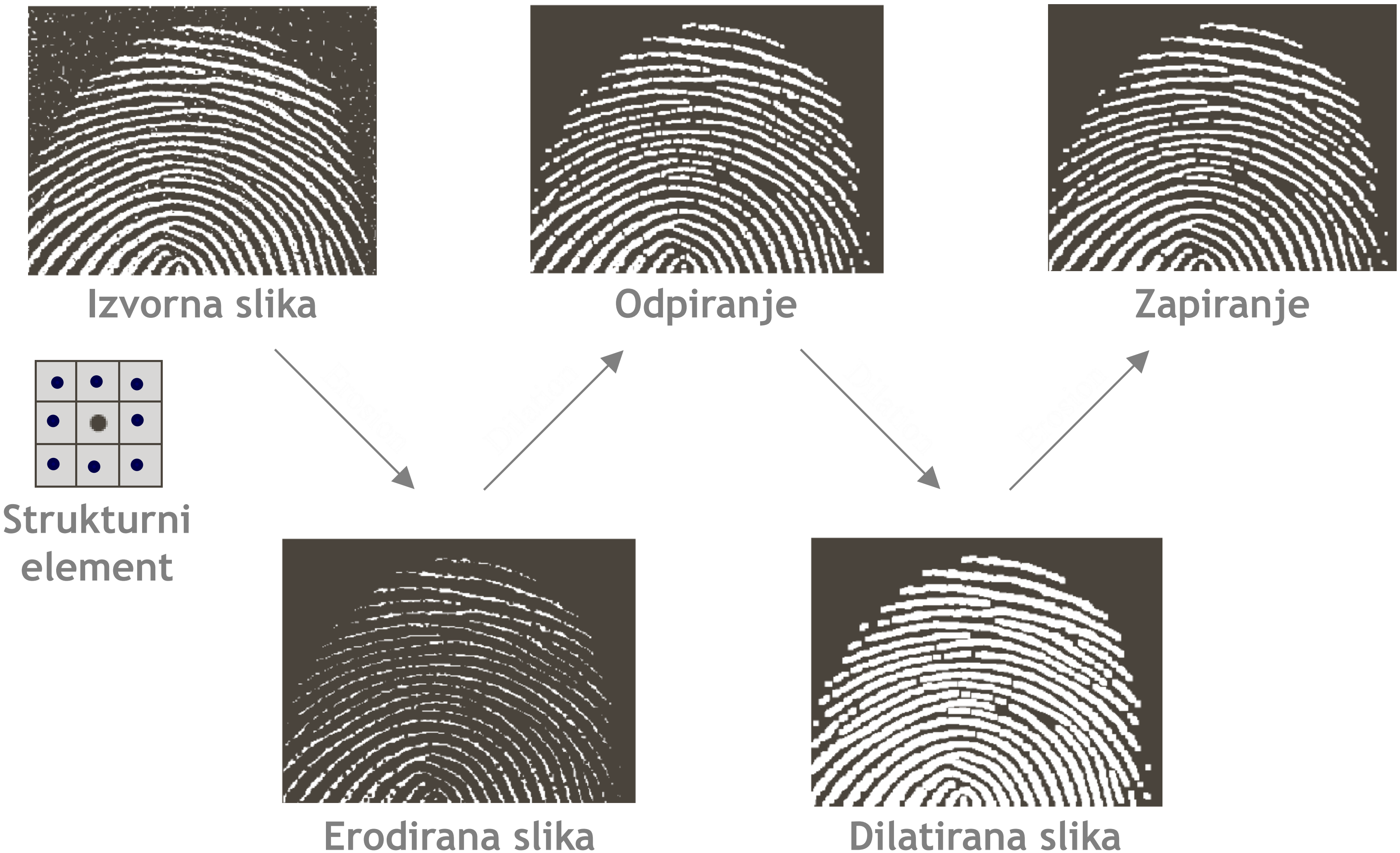
Velikost strukturnega elementa določa velikost lukenj, ki jih bomo zapolnili.





Vir slik: http://homepages.inf.ed.ac.uk/rbf/HIPR2/

# Primer uporabe: odpiranje-zapiranje




Vir slik: R.C. Gonzales & R.E. Woods

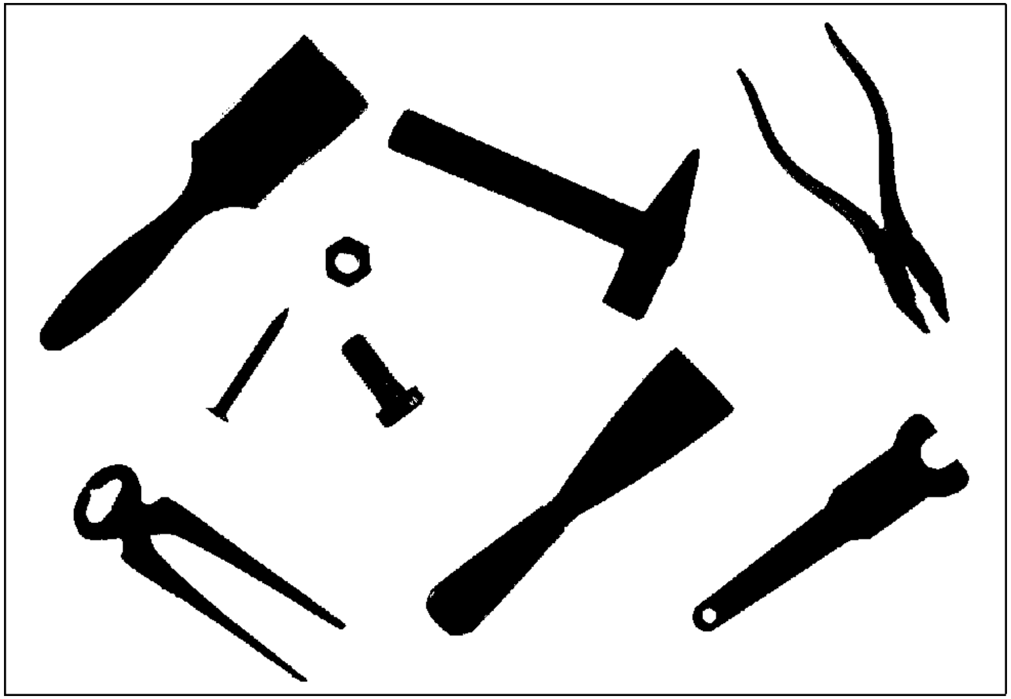
# Morfološki operatorji na sivinskih slikah

- Širitev in Erozija sta tudi posplošeni za uporabo na sivinskih slikah

  - Primer:

# Regije v binarnih slikah

- Ospredje (foreground)
- Ozadje (background)

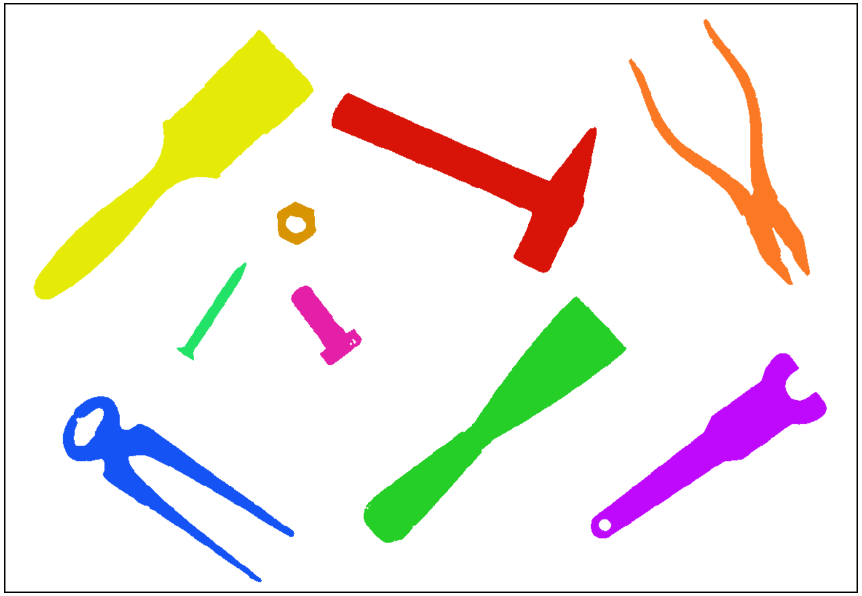


- Število in vrsta predmetov na binarnih slikah

# Iskanje regij na sliki

- Labeliranje (barvanje) regij
  - Kateri slikovni elementi pripadajo kateri regiji?
  - Koliko regij je na sliki?
  - Kje so regije locirane?

$$I(u,v) = \begin{cases} 0 & \textit{background} \text{ pixel} \\ 1 & \textit{foreground} \text{ pixel} \\ 2, 3, \ldots & \text{region } \textit{label}. \end{cases}$$



- Iskanje povezanih slikovnih elementov
  - 4-sosednost
  - 8-sosednost
- Več algoritmov:
  - Poplavljanje (flood filling)
  - Zaporedno označevanje regij (Sequential region marking)
  - Kombinacija obeh

# Barvanje regij s poplavljanjem

- Enostaven algoritem:
  1. Poišči en neoznačen slikovni element ospredja
  2. Označi vse sosednje slikovne elemente ospredja in tako naprej
- Kot poplavljanje pokrajine

- Različni načini poplavljanja:
  1. Rekurzivno
     - zelo prostorosko zahtevno
  2. Iterativno z uporabo sklada (najprej v globino)
  3. Iterativno z uporabo vrste (najprej v širino)
     - najbolj priporočljivo

# Rekurzivni algoritem

```
RegionLabeling(I)
        I: binary image (0 = background, 1 = foreground)
        The image I is labeled (destructively modified) and returned.

    Initialize m ← 2 (the value of the next label to be assigned).
    Iterate over all image coordinates (u, v).
        if I(u, v) = 1 then
            FloodFill(I, u, v, m)        ▷ use any of the 3 versions below
            m ← m + 1.
    return the labeled image I.
```

---

```
FloodFill(I, u, v, label)                          ▷ Recursive Version
    if coordinate (u, v) is within image boundaries and I(u, v) = 1 then
        Set I(u, v) ← label
        FloodFill(I, u+1, v, label)
        FloodFill(I, u, v+1, label)
        FloodFill(I, u, v−1, label)
        FloodFill(I, u−1, v, label)
    return.
```

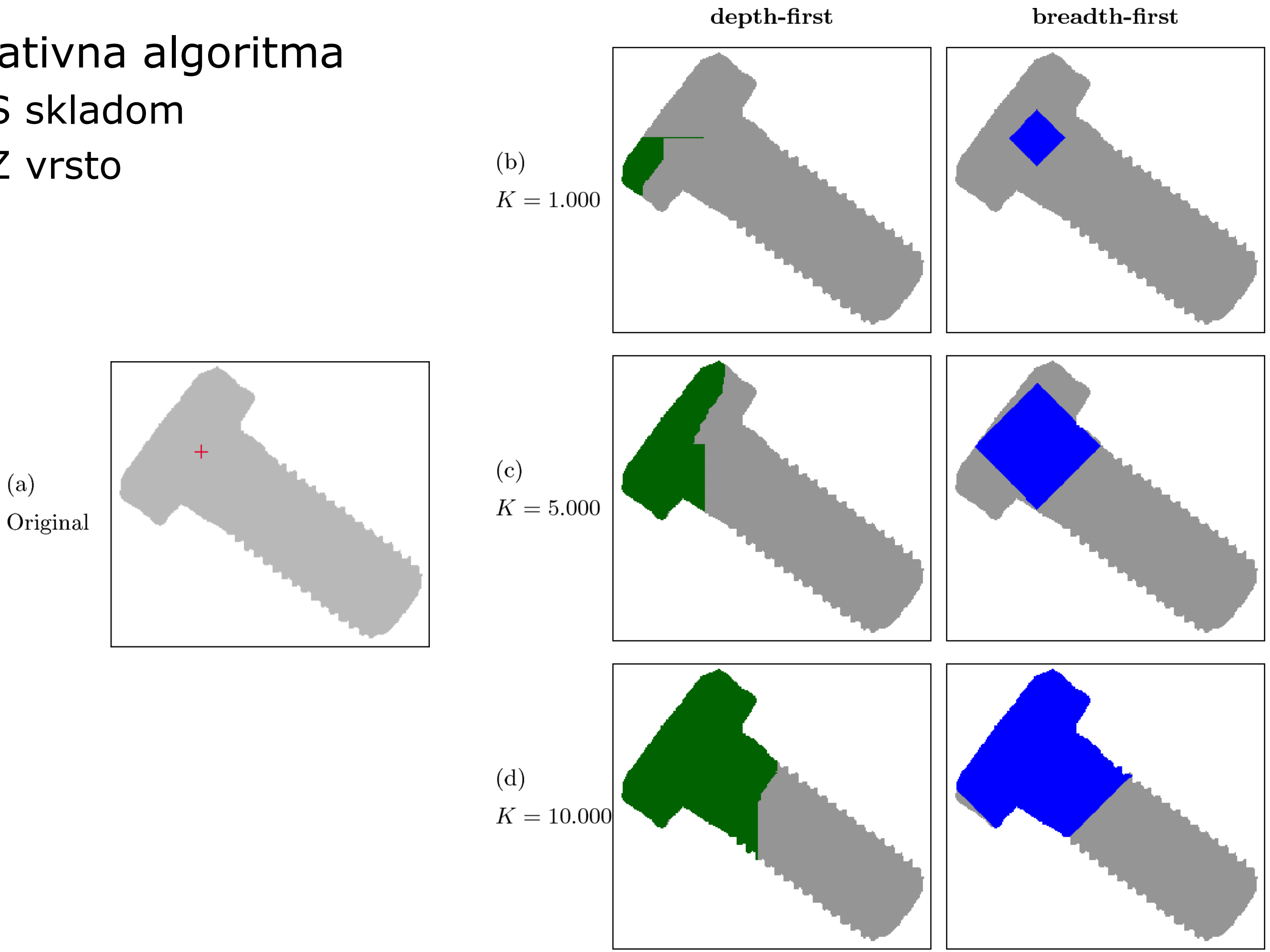
# Iterativna algoritma

```
FloodFill(I, u, v, label)                                   ▷ Depth-First Version
    Create an empty stack S
    Put the seed coordinate ⟨u, v⟩ onto the stack: Push(S, ⟨u, v⟩)
    while S is not empty do
        Get the next coordinate from the top of the stack:
            ⟨x, y⟩ ← Pop(S)
        if coordinate (x, y) is within image boundaries and I(x, y) = 1
            then
            Set I(x, y) ← label
            Push(S, ⟨x+1, y⟩)
            Push(S, ⟨x, y+1⟩)
            Push(S, ⟨x, y−1⟩)
            Push(S, ⟨x−1, y⟩)
    return.

FloodFill(I, u, v, label)                                   ▷ Breadth-First Version
    Create an empty queue Q
    Insert the seed coordinate ⟨u, v⟩ into the queue: Enqueue(Q, ⟨u, v⟩)
    while Q is not empty do
        Get the next coordinate from the front of the queue:
            ⟨x, y⟩ ← Dequeue(Q)
        if coordinate ⟨x, y⟩ is within image boundaries and I(x, y) = 1
            then
            Set I(x, y) ← label
            Enqueue(Q, ⟨x+1, y⟩)
            Enqueue(Q, ⟨x, y+1⟩)
            Enqueue(Q, ⟨x, y−1⟩)
            Enqueue(Q, ⟨x−1, y⟩)
    return.
```

# Primer

- Iterativna algoritma
  - S skladom
  - Z vrsto



# Zaporedno označevanje regij

- Sliko sprocesiramo zaporedno od zgornjega levega do spodnjega desnega vogala v dveh korakih
- Korak 1:
  - Če je trenutni slikovni element del ospredja preverimo del njegovih sosedov

$$\mathcal{N}_4(u,v) = \begin{array}{|c|c|c|} \hline & N_2 & \\ \hline N_1 & \times & \\ \hline & & \\ \hline \end{array} \qquad \text{or} \qquad \mathcal{N}_8(u,v) = \begin{array}{|c|c|c|} \hline N_2 & N_3 & N_4 \\ \hline N_1 & \times & \\ \hline & & \\ \hline \end{array}$$

    - Če so vsi sosedi del ozadja, inicializiramo novo labelo
    - Če imajo vsi ospredni sosedi enako labelo jo priredimo tudi trenutnemu slikovnemu elementu
    - Če imajo različne labele, si zapomnimo to kot konflikt
      - Gradimo grafe med seboj povezanih label
- Korak 2:
  - Razrešimo konflikte: poiščemo povezane komponente v grafih konfliktov in popravimo labelo ustreznih slikovnih elementov

# Primer

(a)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0 Background

1 Foreground

(b) only background neighbors

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

new label (**2**)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Primer

(c) exactly one neighbor label

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

neighbor label is propagated

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(d) two different neighbor labels

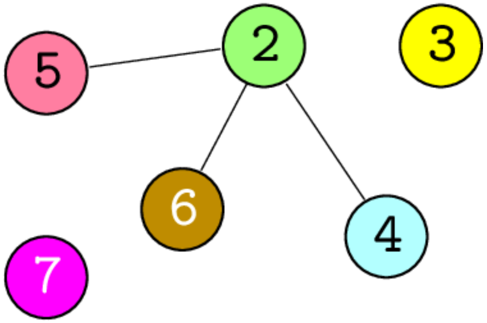
| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 4 | 0 |
| 0 | 5 | 5 | 5 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

one of the labels (**2**) is propagated

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 4 | 0 |
| 0 | 5 | 5 | 5 | 2 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Primer

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 4 | 0 |
| 0 | 5 | 5 | 5 | 2 | 2 | 2 | 0 | 0 | 3 | 0 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 4 | 0 |
| 0 | 6 | 6 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 |
| 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 |
| 0 | 7 | 7 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |




| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 2 | 0 |
| 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 3 | 0 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 |
| 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 |
| 0 | 7 | 7 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Algoritem – korak 1

SequentialLabeling($I$)

$I$: binary image ($0 = \textit{background}$, $1 = \textit{foreground}$)

The image $I$ is labeled (destructively modified) and returned.

Pass 1—Assign Initial Labels:

Initialize $m \leftarrow 2$ (the value of the next label to be assigned).

Create an empty set $\mathcal{C}$ to hold the collisions: $\mathcal{C} \leftarrow \{\}$.

**for** $v \leftarrow 0 \ldots H-1$ **do** ▷ $H$ = height of image $I$

  **for** $u \leftarrow 0 \ldots W-1$ **do** ▷ $W$ = width of image $I$

    **if** $I(u,v) = 1$ **then** do one of:

      **if** all neighbors of $(u,v)$ are background pixels (all $n_i = 0$) **then**

        $I(u,v) \leftarrow m$.

        $m \leftarrow m+1$.

      **else if** exactly *one* of the neighbors has a label value $n_k > 1$ **then**

        set $I(u,v) \leftarrow n_k$

      **else if** *several* neighbors of $(u,v)$ have label values $n_j > 1$ **then**

        Select one of them as the new label: $I(u,v) \leftarrow k \in \{n_j\}$.

        **for** all other neighbors of $u,v)$ with label values $n_i > 1$ and $n_i \neq k$ **do**

          Create a new label collision $\boldsymbol{c}_i = \langle n_i, k \rangle$.

          Record the collision: $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{c}_i\}$.

*Remark:* The image $I$ now contains label values $0, 2, \ldots m-1$.

# Algoritem – korak 2

PASS 2—RESOLVE LABEL COLLISIONS:

Let $\mathcal{L} = \{2, 3, \ldots m-1\}$ be the set of preliminary region labels.

Create a partitioning of $\mathcal{L}$ as a *vector of sets*, one set for each label value: $\mathcal{R} \leftarrow [\mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_{m-1}] = [\{2\}, \{3\}, \{4\}, \ldots, \{m-1\}]$, so $\mathcal{R}_i = \{i\}$ for all $i \in \mathcal{L}$.

**for all** collisions $\langle a, b\rangle \in \mathcal{C}$ **do**

    Find in $\mathcal{R}$ the sets $\mathcal{R}_a$, $\mathcal{R}_b$ containing the labels $a$, $b$, resp.:

        $\mathcal{R}_a \leftarrow$ the set that currently contains label $a$

        $\mathcal{R}_b \leftarrow$ the set that currently contains label $b$

    **if** $\mathcal{R}_a \neq \mathcal{R}_b$ ($a$ and $b$ are contained in different sets) **then**

        Merge sets $\mathcal{R}_a$ and $\mathcal{R}_b$ by moving all elements of $\mathcal{R}_b$ to $\mathcal{R}_a$:

            $\mathcal{R}_a \leftarrow \mathcal{R}_a \cup \mathcal{R}_b$

            $\mathcal{R}_b \leftarrow \{\}$

*Remark:* All *equivalent* label values (i. e., all labels of pixels in the same region) are now contained in the same set $\mathcal{R}_i$ within $\mathcal{R}$.

PASS 3—RELABEL THE IMAGE:

Iterate through all image pixels $(u, v)$:

    **if** $I(u, v) > 1$ **then**

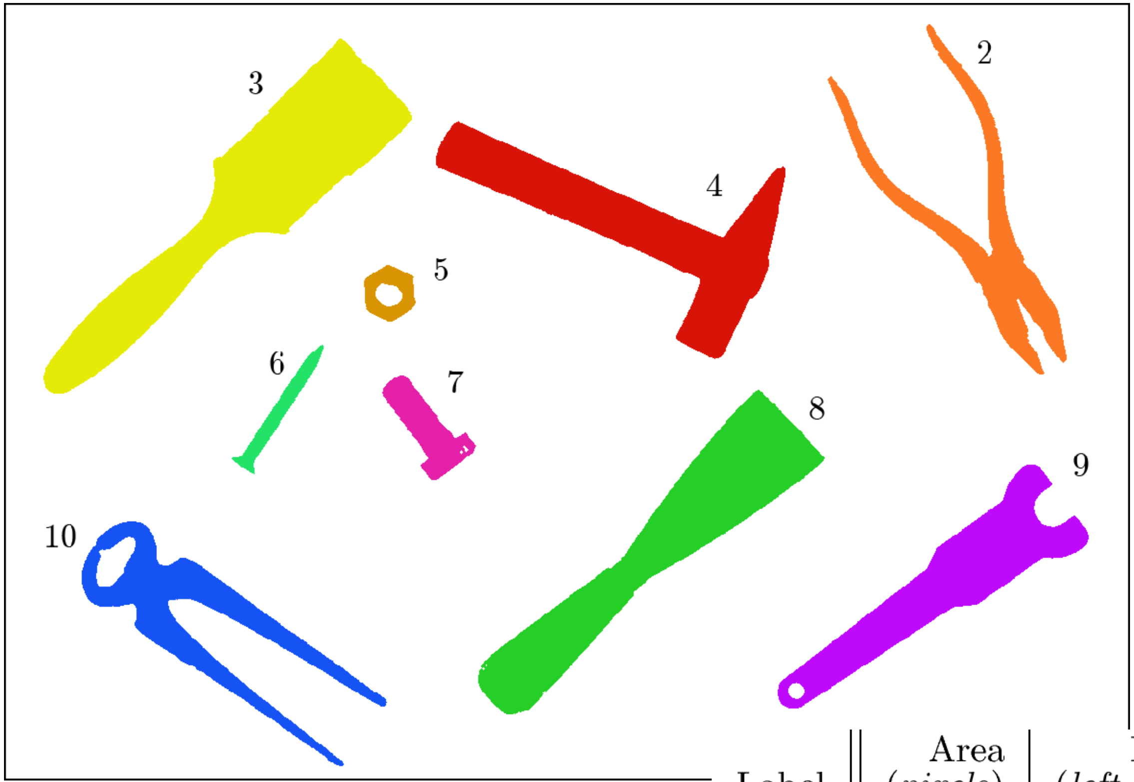
        Find the set $\mathcal{R}_i$ in $\mathcal{R}$ that contains label $I(u, v)$.

        Choose one unique representative element $k$ from the set $\mathcal{R}_i$ (e. g., the minimum value, $k = \min(\mathcal{R}_i)$).

        Replace the image label: $I(u, v) \leftarrow k$.

**return** the labeled image $I$.

# Primer

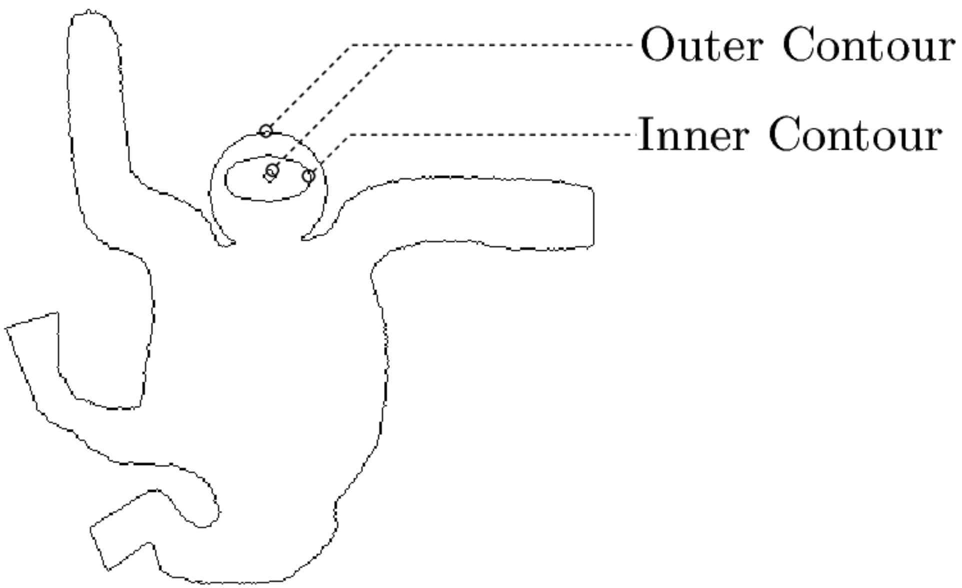



| Label | Area (*pixels*) | Bounding Box (*left*, *top*, *right*, *bottom*) | Center $(x_c, y_c)$ |
|---|---|---|---|
| 2 | 14978 | (887, 21, 1144, 399) | (1049.7, 242.8) |
| 3 | 36156 | ( 40, 37, 438, 419) | ( 261.9, 209.5) |
| 4 | 25904 | (464, 126, 841, 382) | ( 680.6, 240.6) |
| 5 | 2024 | (387, 281, 442, 341) | ( 414.2, 310.6) |
| 6 | 2293 | (244, 367, 342, 506) | ( 294.4, 439.0) |
| 7 | 4394 | (406, 400, 507, 512) | ( 454.1, 457.3) |
| 8 | 29777 | (510, 416, 883, 765) | ( 704.9, 583.9) |
| 9 | 20724 | (833, 497, 1168, 759) | (1016.0, 624.1) |
| 10 | 16566 | ( 82, 558, 411, 821) | ( 208.7, 661.6) |

# Obrisi regij

- Morfološki operatorji označijo slikovne elemente na obrisu
- Potrebujemo tudi urejeno zaporedje robnih slikovnih elementov

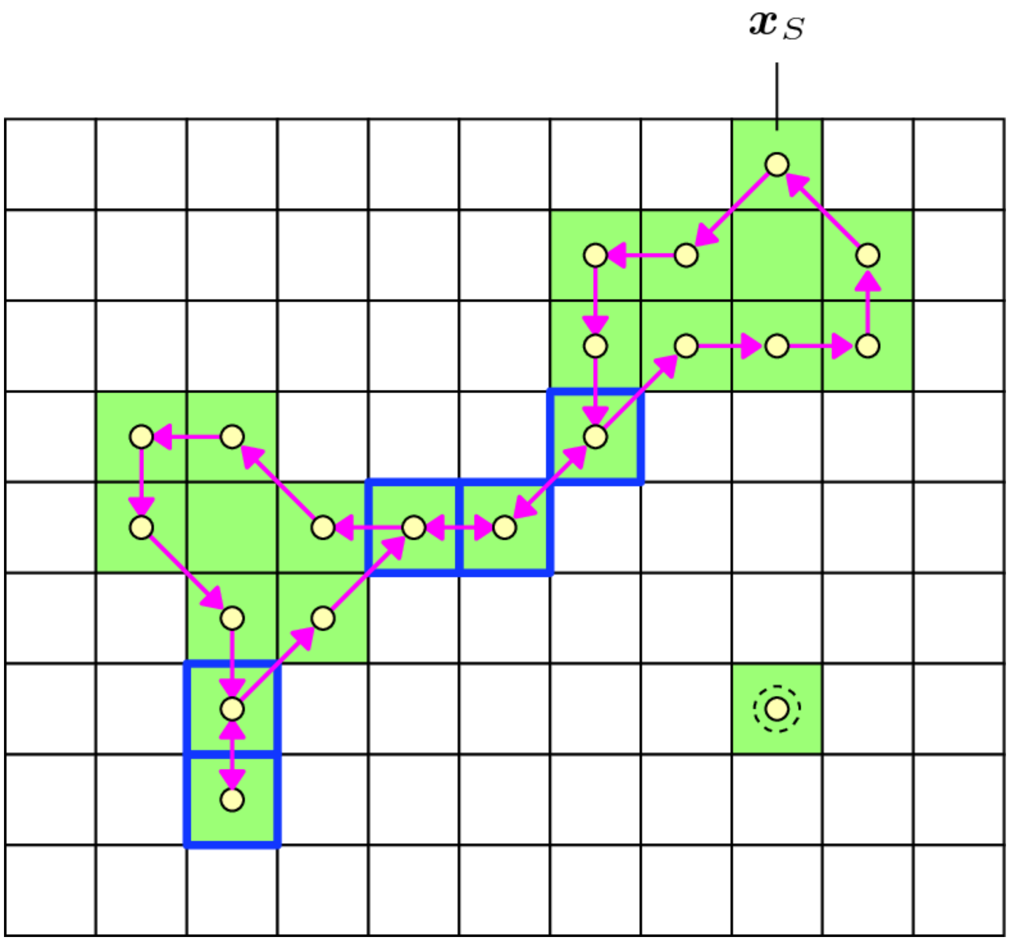
- Zunanji obris in notranji obris



# Iskanje obrisa

1. Poišči povezano regijo
2. Poišči rob regije in mu sledi naokrog

- Težave:
  - Notranji obrisi
  - Deli regij debeline en sl. element
  - Izolirani s- elementi



# Kombinirano iskanje regij in obrisov

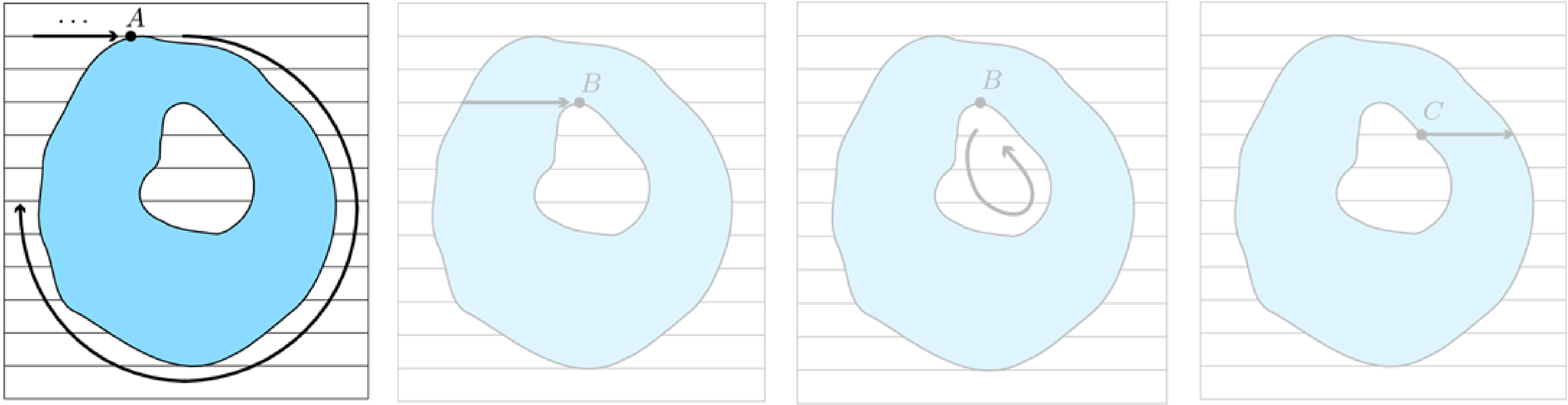
- Hkrati označi regije in poišče obrise

- Algoritem preišče celotno sliko od zgornjega levega do spodnjega desnega slikovnega elementa
- V nekem slikovnem elementu so možni trije primeri:
  - Primer A: BG->neoznačen FG => zunanji obris
    - Dodeli novo oznako regiji
    - Prepotuj celoten zunanji obris in ustrezno označi sl. elemente
    - Sosednje BG sl. elemente označi z -1
  - Primer B: FG->neoznačen BG => notranji obris
    - Prepotuj celoten notranji obris in ustrezno označi sl. elemente
    - Mejne BG sl. elemente označi z -1
  - Primer C: označen FG
    - Prenesi oznako na sl. elemente na desni

# Kombinirano iskanje regij in obrisov

- Hkrati označi regije in poišče obrise

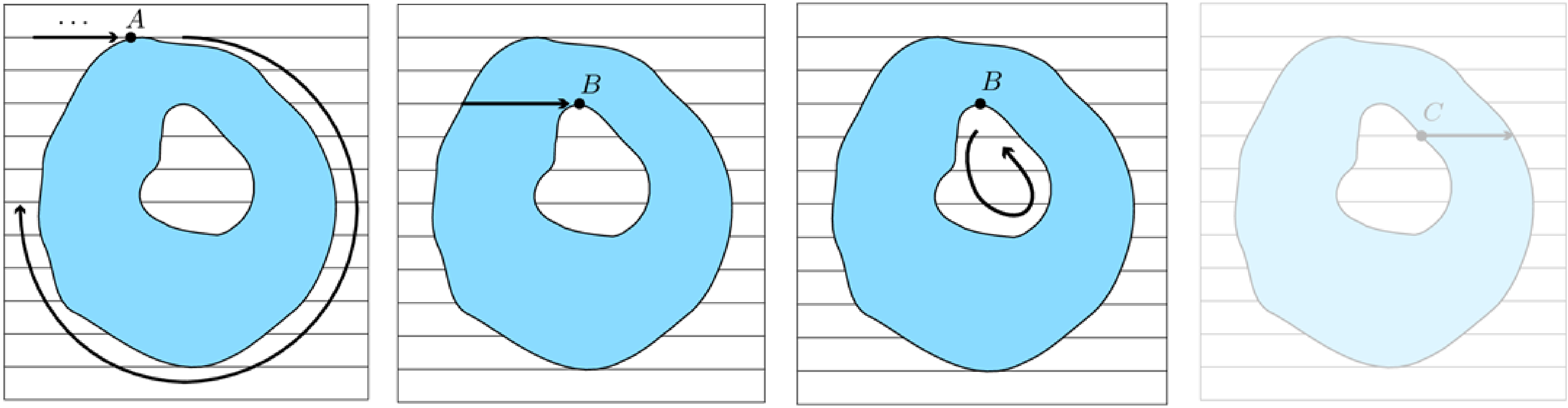
- Algoritem preišče celotno sliko od zgornjega levega do spodnjega desnega slikovnega elementa
- V nekem slikovnem elementu so možni trije primeri:
  - **Primer A**: BG->neoznačen FG => zunanji obris
    - Dodeli novo oznako regiji
    - Prepotuj celoten zunanji obris in ustrezno označi sl. elemente
    - Sosednje BG sl. elemente označi z -1



# Kombinirano iskanje regij in obrisov

- Hkrati označi regije in poišče obrise

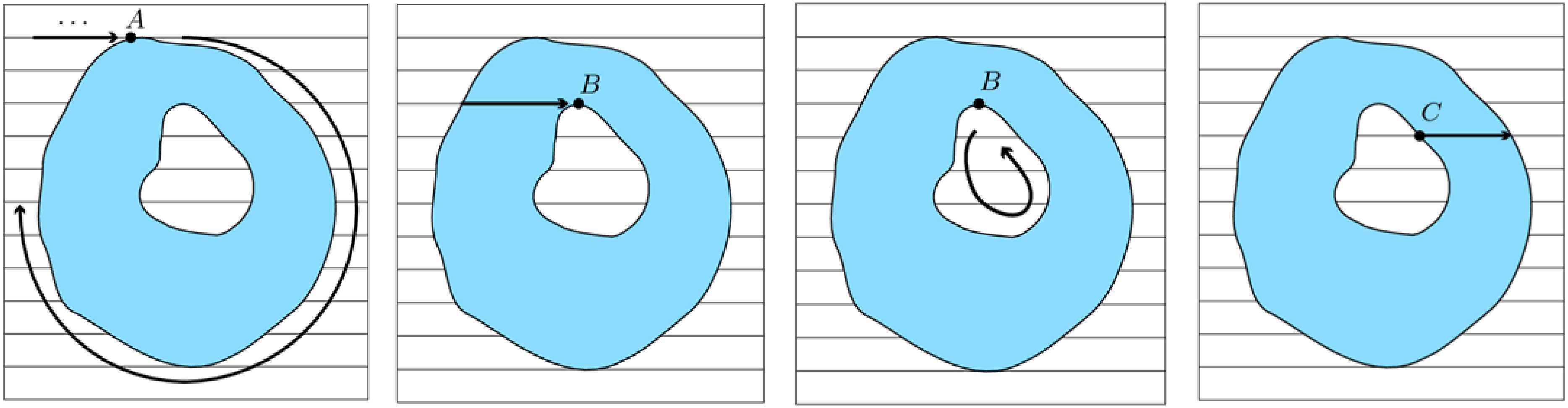
- Algoritem preišče celotno sliko od zgornjega levega do spodnjega desnega slikovnega elementa
- V nekem slikovnem elementu so možni trije primeri:
  - **Primer B**: FG->neoznačen BG => notranji obris
    - Prepotuj celoten notranji obris in ustrezno označi sl. elemente
    - Mejne BG sl. elemente označi z -1



# Kombinirano iskanje regij in obrisov

- Hkrati označi regije in poišče obrise

- Algoritem preišče celotno sliko od zgornjega levega do spodnjega desnega slikovnega elementa
- V nekem slikovnem elementu so možni trije primeri:
  - **Primer C**: prehod iz FG na neoznačen FG
    - Prenesi oznako na sl. elemente na desni



# Algoritem

```
CombinedContourLabeling (I)
        I: binary image
        Returns a set of contours and a label map (labeled image).
    Create an empty set of contours: C ← {}
    Create a label map LM of the same size as I and initialize:
    for all (u, v) do
        LM(u, v) ← 0                                   ▷ label map LM
    R ← 0                                              ▷ region counter R

    Scan the image from left to right and top to bottom:
    for v ← 0 ... N−1 do
        Lk ← 0                                         ▷ current label Lk
        for u ← 0 ... M−1 do
            if I(u, v) is a foreground pixel then
                if (Lk ≠ 0) then                       ▷ continue existing region
                    LM(u, v) ← L
                else
                    Lk ← LM(u, v)
                    if (Lk = 0) then                   ▷ hit new outer contour
                        R ← R + 1
                        Lk ← R
                        xS ← (u, v)
                        c_outer ← TraceContour(xS, 0, Lk, I, LM)
                        C ← C ∪ {c_outer}              ▷ collect new contour
                        LM(u, v) ← Lk
            else                                       ▷ I(u, v) is a background pixel
                if (L ≠ 0) then
                    if (LM(u, v) = 0) then             ▷ hit new inner contour
                        xS ← (u−1, v)
                        c_inner ← TraceContour(xS, 1, Lk, I, LM)
                        C ← C ∪ {c_inner}              ▷ collect new contour
                    L ← 0
    return (C, LM).             ▷ return the set of contours and the label map
```

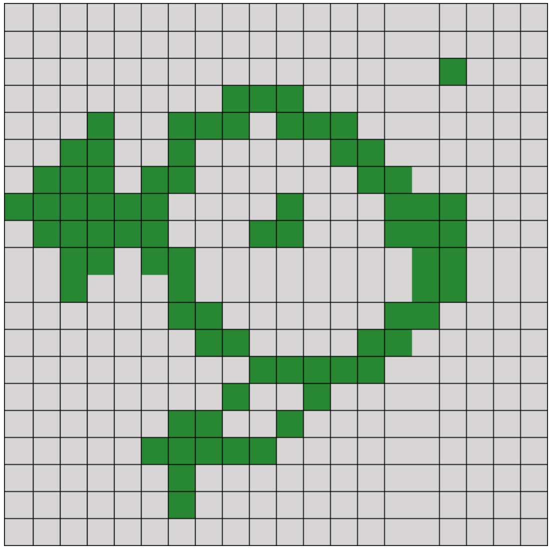
```
TraceContour(xS, dS, Lk, I, LM)
        xS: start position, dS: initial search direction,
        Lc: label for this contour
        I: original image, LM: label map.
        Traces and returns the contour starting at xS.
    (xT, d_next) ← FindNextPoint(xS, dS, I, LM)
    c ← [xT]                                ▷ create a contour starting with xT
    xp ← xS                                 ▷ previous position xp = (up, vp)
    xc ← xT                                 ▷ current position xc = (uc, vc)
    done ← (xS ≡ xT)                        ▷ isolated pixel?
    while (¬done) do
        LM(uc, vc) ← Lc
        d_search ← (d_next + 6) mod 8
        (xn, d_next) ← FindNextPoint(xc, d_search, I, LM)
        xp ← xc
        xc ← xn
        done ← (xp ≡ xS ∧ xc ≡ xT)          ▷ back at start point?
        if (¬done) then
            Append(c, xn)                   ▷ add point xn to contour c
    return c.                               ▷ return this contour
```

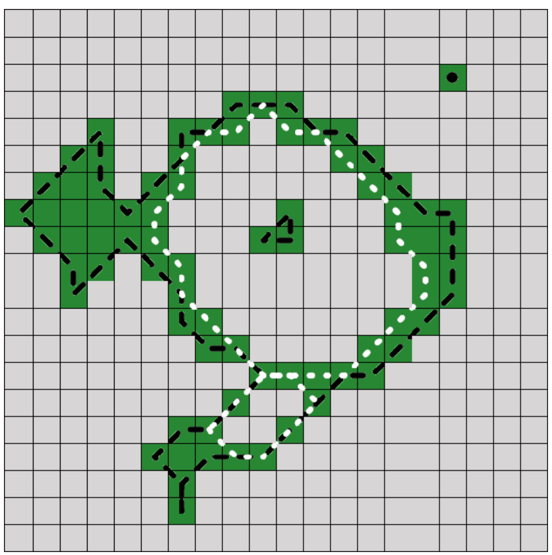
```
FindNextPoint(xc, d, I, LM)
        xc: start point, d: search direction,
        I: original image, LM: label map.
    for i ← 0 ... 6 do                      ▷ search in 7 directions
        x′ ← xc + Delta(d)                  ▷ x′ = (u′, v′)
        if I(u′, v′) is a background pixel then
            LM(u′, v′) ← −1                 ▷ mark background as visited (−1)
            d ← (d + 1) mod 8
        else                                ▷ found a nonbackground pixel at x′
            return (x′, d)
    return (xc, d).                         ▷ found no next point, return start point
```

Delta(d) = (Δx, Δy), with

| $d$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $\Delta x$ | 1 | 1 | 0 | −1 | −1 | −1 | 0 | 1 |
| $\Delta y$ | 0 | 1 | 1 | 1 | 0 | −1 | −1 | −1 |

# Primer



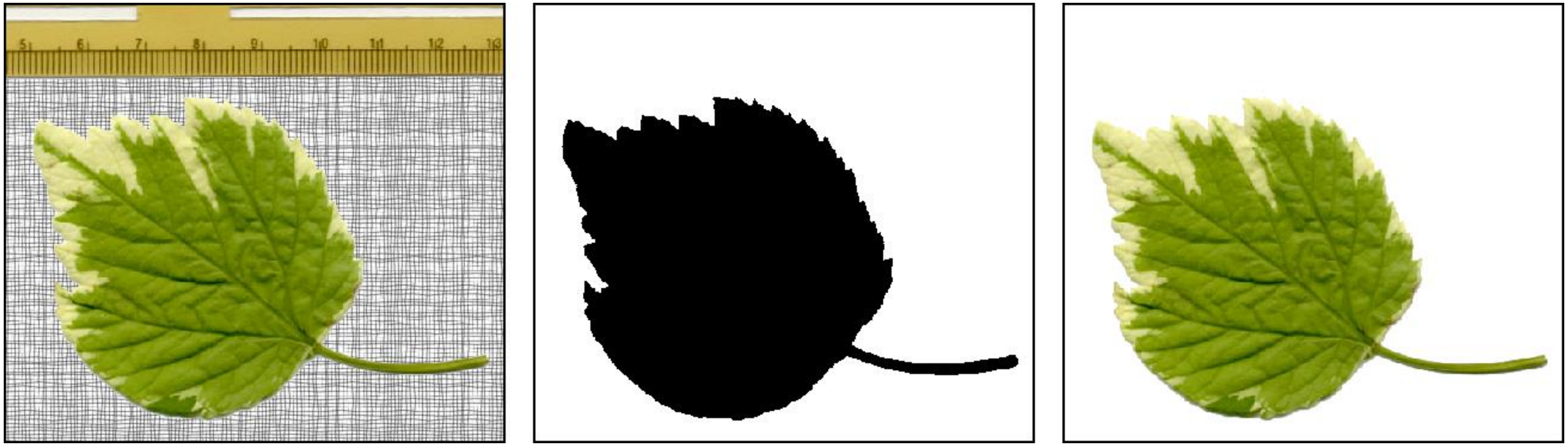


# Predstavitve slikovnih regij

- Predstavitev z matriko
- Run Length Encoding (RLE)
- Absolutna verižna koda
- Diferenčna verižna koda

# Predstavitev z matriko

- Najbolj klasična predstavitev
- Binarna maska določa regijo:



- Predstavitev ni odvisna vsebine slike
- Pogosto neučinkovita (prostorsko potratna)

# Run Length Encoding (RLE)

- Kodiranje zaporedij slikovnih elementov ospredja s tremi parametri:

$$Run_i = \langle \mathsf{row}_i, \mathsf{column}_i, \mathsf{length}_i \rangle$$

- Primer:

Bitmap

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | |
| 1 | | | × | × | × | × | × | × | |
| 2 | | | | | | | | | |
| 3 | | | | | × | × | × | × | |
| 4 | | × | × | × | | × | × | × | |
| 5 | × | × | × | × | × | × | × | × | × |
| 6 | | | | | | | | | |

$\longrightarrow$

RLE

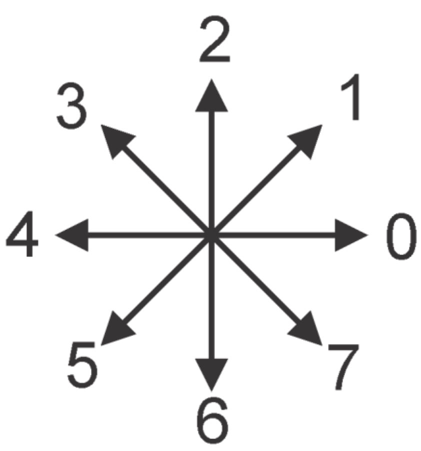
$\langle \mathsf{row}, \mathsf{column}, \mathsf{length} \rangle$

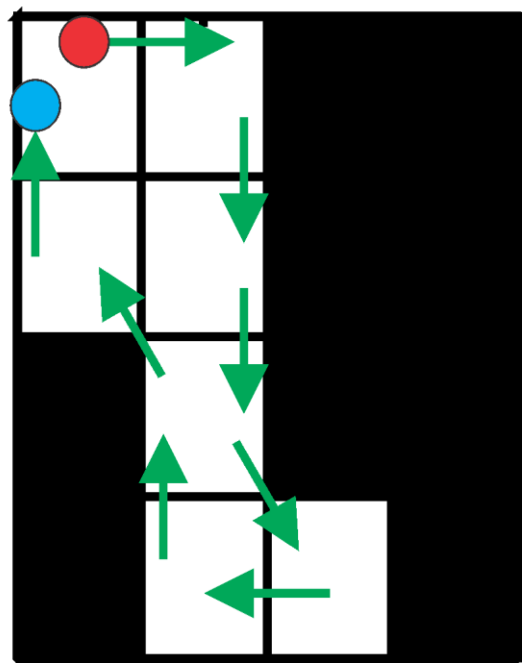
$\langle 1, 2, 6 \rangle$
$\langle 3, 4, 4 \rangle$
$\langle 4, 1, 3 \rangle$
$\langle 4, 5, 3 \rangle$
$\langle 5, 0, 9 \rangle$

- Prvi parameter lahko spustimo, če je vrstica ista
- Enostavna brezizgubna metoda za kompresijo
- Že zelo dolgo, veliko uporabljana (faks, algoritmi za kompresijo slik, itd.)

# Kodiranje obrisa z verižno kodo

- Absolute Chain Code, tudi Freemanove kode
- Zapis diskretizirane oblike z eno samo številko
- Sprehodimo se po obrisu regije in vsak premik zakodiramo s smerjo v katero smo se premaknili



Absolutna koda:
06674232

# Absolutna verižna koda

- Kodiranje obrisa regije

- Začnemo v nekem robnem slikovnem elementu in gremo po obrisu, pri čemer vsak korak zakodiramo:

$$\boldsymbol{c}_{\mathcal{R}} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots \boldsymbol{x}_{M-1}] \text{ with } \boldsymbol{x}_i = \langle u_i, v_i \rangle$$

$$\boldsymbol{c}'_{\mathcal{R}} = [c'_0, c'_1, \ldots c'_{M-1}]$$

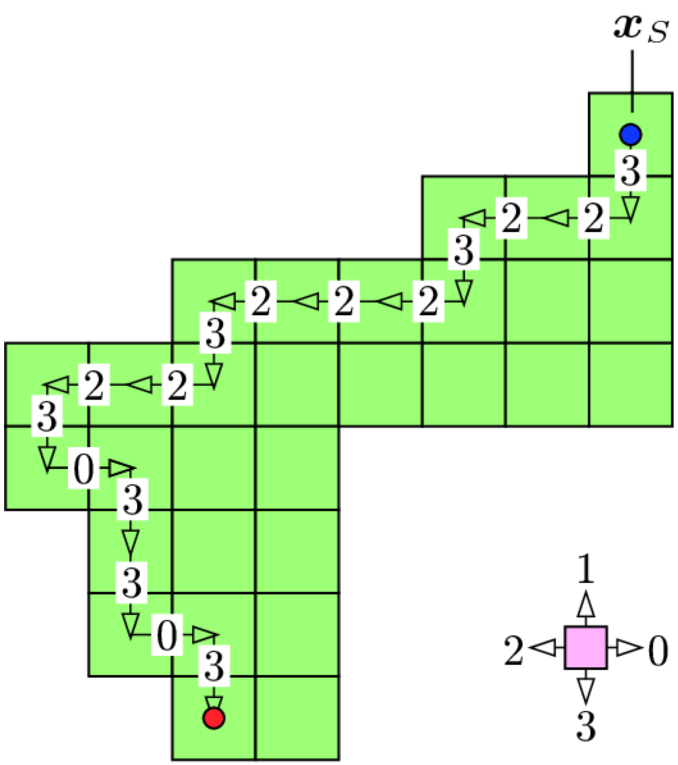
$$c'_i = \textsc{Code}(\Delta u_i, \Delta v_i)$$

$$(\Delta u_i, \Delta v_i) = \begin{cases} (u_{i+1} - u_i, v_{i+1} - v_i) & \text{for } 0 \leq i < M-1 \\ (u_0 - u_i, v_0 - v_i) & \text{for } i = M-1, \end{cases}$$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta u$ | 1 | 1 | 0 | −1 | −1 | −1 | 0 | 1 |
| $\Delta v$ | 0 | 1 | 1 | 1 | 0 | −1 | −1 | −1 |
| $\textsc{Code}(\Delta u, \Delta v)$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

# Primer

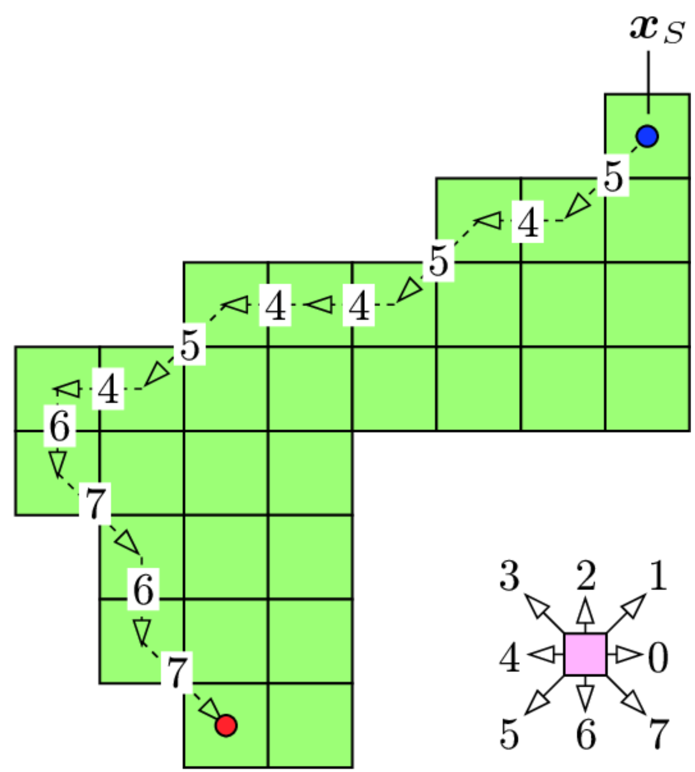
- Lahko upoštevamo 4- ali 8-sosednost




**4-Chain Code**

322322232303303...111

$length = 28$




**8-Chain Code**

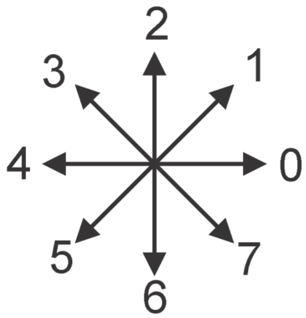
54544546767...222

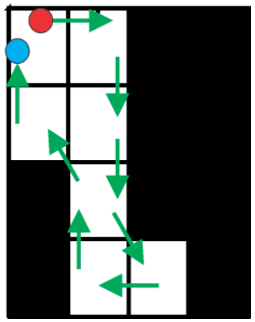
$length = 16 + 6\sqrt{2} \approx 24{,}5$

- Koda je odvisna od začetne točke in orientacije obrisa

# Absolutna vs. diferenčna verižna koda

- Koda ni invariantna na rotacijo, zato izračunamo nekakšne absolutne „odvode“ – *Diferenčna Freemanova koda*:
  - Izračunamo razlike med zaporednimi številkami kode
  - Razlike delimo po modulu števila smeri v tarči



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| koda: 06674232 | | | | | | | | |
| diferenčna koda: | (6-0) | (6-6) | (7-6) | (4-7) | (2-4) | (3-2) | (2-3) | (0-2) |
| | 6 | 0 | 1 | -3 | -2 | 1 | -1 | -2 |
| modulo 8: | 6 | 0 | 1 | 5 | 6 | 1 | 7 | 6 |

# Diferenčna verižna koda

- Differential chain code
- Namesto kodiranja razlike pozicij, kodiramo razlike smeri

$$c''_i = \begin{cases} (c'_{i+1} - c'_i) \bmod 8 & \text{for } 0 \leq i < M-1 \\ (c'_0 - c'_i) \bmod 8 & \text{for } i = M-1 \end{cases}$$

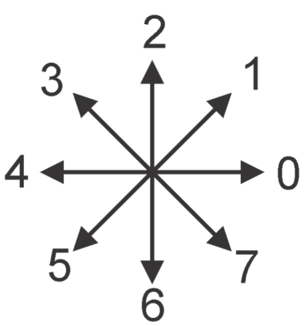
- Primer:

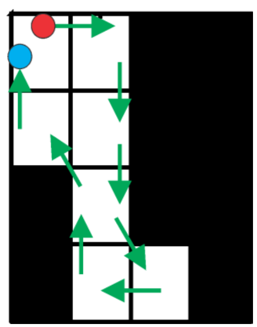
$$\boldsymbol{c}'_{\mathcal{R}} = [5, 4, 5, 4, 4, 5, 4, 6, 7, 6, 7, \ldots 2, 2, 2]$$
$$\boldsymbol{c}''_{\mathcal{R}} = [7, 1, 7, 0, 1, 7, 2, 1, 7, 1, 1, \ldots 0, 0, 3]$$

- Sedaj lahko regijo zarotiramo za 90 stopinj, pa se koda ne spremeni
- Še vedno pa je koda odvisna od začetne pozicije

# Številke oblike

- Diferenčna Freemanova koda ni invariantna na začetek iskanja, zato jo normaliziramo v najnižjo desetiško številko:
  - Kodo rotiramo za tak *d* da dobimo najnižjo (ali najvišjo) desetiško vrednost



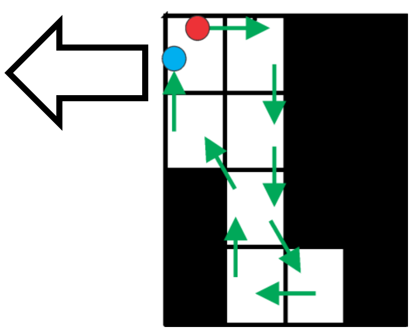



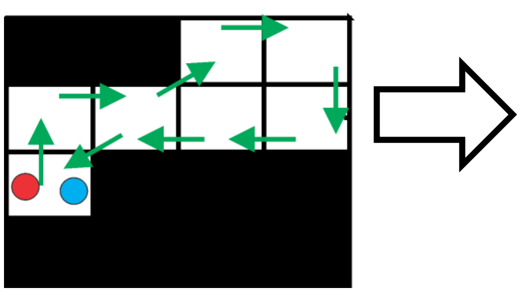
Feemanova koda: 06674232

Diferenčna Freemanova koda: 60156176

Normalizirana diferenčna Freemanova koda: 01561766

NdFK:
01561766





NdFK:
01561766

# Številke oblike

- Shape numbers
- Verižne kode moramo poravnati, da imajo enako začetno točko, potem jih lahko primerjamo
- Elemente kode interpretiramo kot števke v bazi b (8 ali 4):

$$\mathrm{VAL}(\boldsymbol{c}''_{\mathcal{R}}) = c''_0 \cdot b^0 + c''_1 \cdot b^1 + \ \ldots \ = \sum_{i=0}^{M-1} c''_i \cdot b^{i-1}$$

- Poiščemo maksimum (ali min.) in kodo zamaknemo za *k* mest

$$k_{\max} = \arg \max_{0 \leq k < M} \mathrm{VAL}(\boldsymbol{c}''_{\mathcal{R}} \triangleright k)$$

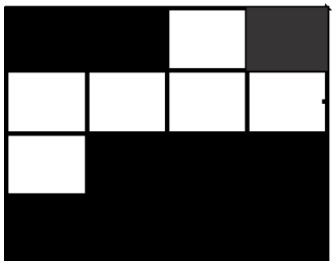
- Ni potrebno dejansko tega računati, lahko samo sortiramo
- Primer:

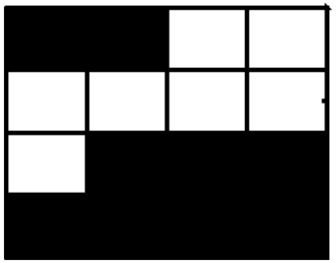
$$\boldsymbol{c}''_{\mathcal{R}} = [\,0, 1, 3, 2, \ldots 9, 3, 7, 4\,]$$
$$\boldsymbol{c}''_{\mathcal{R}} \triangleright 2 = [\,7, 4, 0, 1, 3, 2, \ldots 9, 3\,]$$

- Zdaj koda ni več odvisna od začetnega položaja, lahko je regija rotirana za 90 stopinj
- Še vedno pa predstavitev ni invariantna na poljubne rotacije in skaliranje
  - Rešitev: Fourierjevi deskriptorji

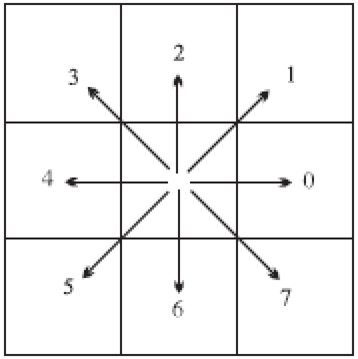
# Verižna koda

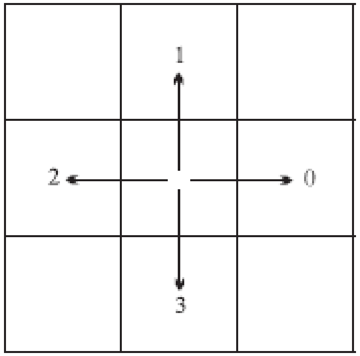
- Kako primerjati dve kodi F1 in F2?
- Vsako kodo obravnavamo kot niz znakov in izračunamo koliko vrivanj znakov in premeščanj znakov bi rabili, da transformiramo kodo F1 v F2.
    - Levenshteinova razdalja





- Z uporabo različnih tarč dobimo različne kode.
    - Osemsmerna tarča
    - Štirismerna tarča



# Lastnosti binarnih regij

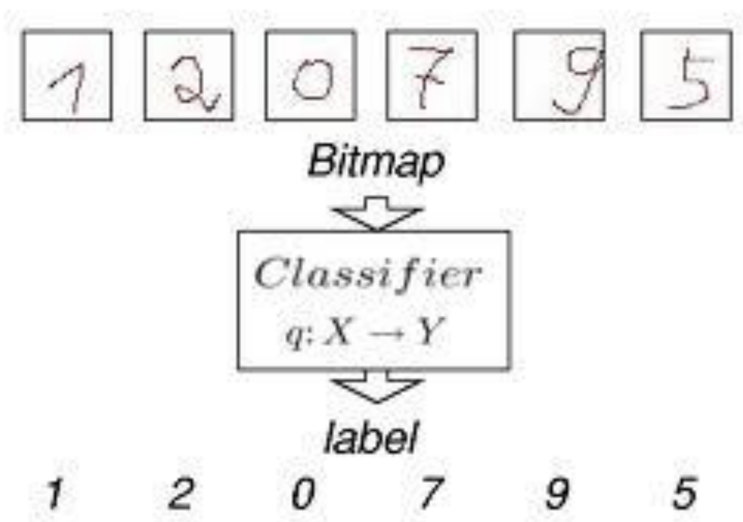
- Kako opisati sliko
  - Matrika števil: zaporedje slikovnih elementov
  - „Pretežno bel krog na pretežno zeleni podlagi."
  - „Nogometna žoga na travi."

- Zelo težko je priti do semantičnega opisa slike
- Lažje posamezne regije na sliki opišemo z enostavnimi lastnostmi

# Značilnice

- Ko imamo regije, jih opišemo z značilnicami
- Na osnovi vektorjev značilnic lahko regije primerjamo med seboj
  - Lahko iščemo podobnosti med regijami
  - Lahko merimo in preverjamo dimenzije in oblike

- Primer: OCR:




# Geometrične značilnice

- Geometrične značilnice definiramo za binarno regijo

$$\mathcal{R} = \{\boldsymbol{x}_0, \boldsymbol{x}_1 \ldots \boldsymbol{x}_{N-1}\} = \{(u_0, v_0), (u_1, v_1) \ldots (u_{N-1}, v_{N-1})\}$$

- Obseg (Perimeter)
- Površina (Area)
- Kompaktnost oz. okroglost (Compactness and roundness)
- Obsegajoč pravokotnik (Bounding box)
- Konveksna ovojnica (Convex hull)

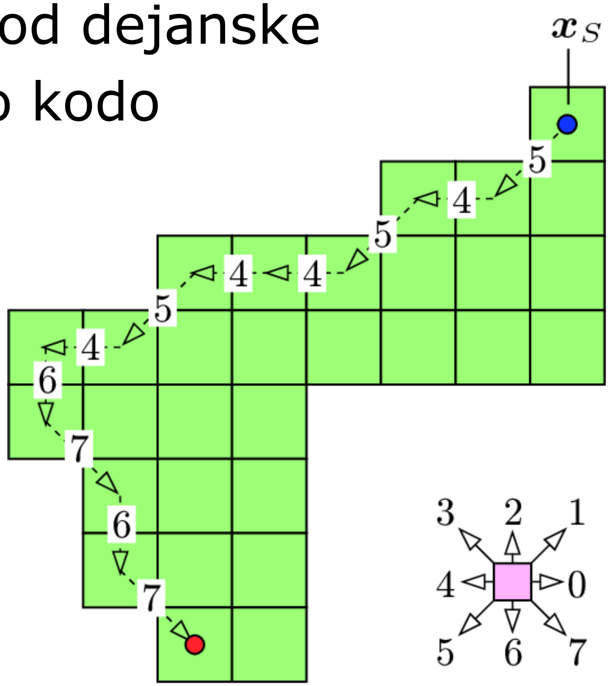
# Obseg

- Perimeter
- Dolžina zunanjega obsega povezane regije
- V primeru 4-sosednosti bo dolžina večja od dejanske
- Ponavadi uporabljamo 8-sosedno Verižno kodo

$$\boldsymbol{c}'_{\mathcal{R}} = [c'_0, c'_1, \ldots c'_{M-1}]$$

$$\text{Perimeter}(\mathcal{R}) = \sum_{i=0}^{M-1} \text{length}(c'_i)$$

$$\text{with} \quad \text{length}(c) = \begin{cases} 1 & \text{for} \quad c = 0, 2, 4, 6 \\ \sqrt{2} & \text{for} \quad c = 1, 3, 5, 7 \end{cases}$$




**8-Chain Code**

54544546767...222

*length* = 16 + $6\sqrt{2} \approx$ 24,5

- To nam vrne nekoliko prevelik obseg
  - Zaradi diskretizacije poševnih ravnih črt
  - Normaliziramo:

$$P(\mathcal{R}) \approx \text{Perimeter}_{\text{corr}}(\mathcal{R}) = 0.95 \cdot \text{Perimeter}(\mathcal{R})$$

# Površina

- Area
- Število slikovnih elementov v regiji:

$$A(\mathcal{R}) = |\mathcal{R}| = N.$$

- Lahko tudi ocenimo iz obrisa
  - če regija nima lukenj
  - Uporabimo zapis v Verižni kodi:

$$A(\mathcal{R}) \approx \frac{1}{2} \cdot \left| \sum_{i=0}^{M-1} \left( u_i \cdot v_{(i+1) \bmod M} \; - \; u_{(i+1) \bmod M} \cdot v_i \right) \right|$$

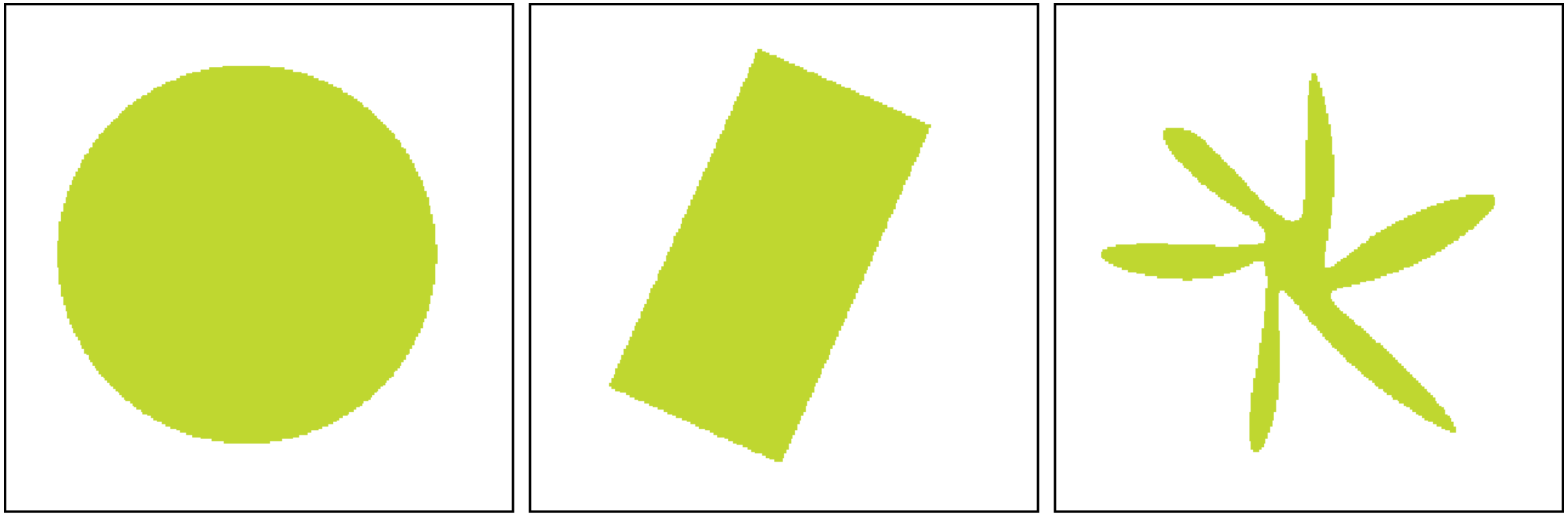
- Tako obseg kot površina sta
  - Neodvisna od translacije in rotacije
  - Odvisna od skale (velikosti, oddaljenosti)

# Kompaktnost oz. okroglost

- Compactness and circularity
- Meri kako kompaktna oz. okrogla je regija

$$\text{Circularity}(\mathcal{R}) = 4\pi \cdot \frac{A(\mathcal{R})}{P^2(\mathcal{R})}$$

- Invariantna na translacijo, rotacijo in skalo
- Primer:

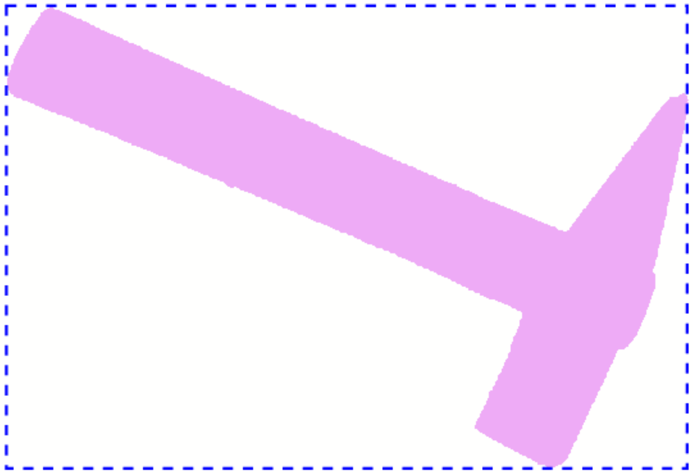


1.001 0.672 0.086

# Obsegajoč pravokotnik

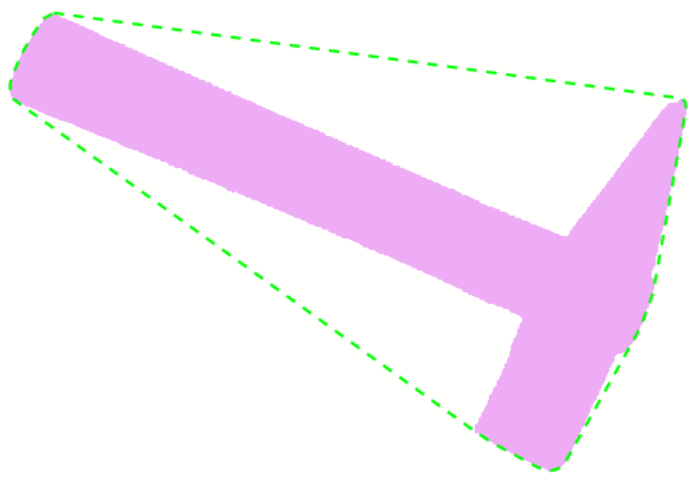
- Bounding box
- Najmanjši pravokotnik z osmi vzporednimi s koordinatnimi osmi, ki vsebuje celotno regijo

$$\mathsf{BoundingBox}(\mathcal{R}) = \langle u_{\min}, u_{\max}, v_{\min}, v_{\max} \rangle$$



# Konveksna ovojnica

- Convex hull
- Najmnjši poligon, ki vsebuje vse elemente regije
- Računamo jo lahko z algoritmom QuickHull (kompleksnost O(NH) )



- Konveksnost = dolžina konveksne ovojnice / obseg regije
- Gostota = površina regije / površina konveksne ovojnice
- Premer = razdalja me dvema maksimalno oddaljenima
  točkama na konveksni ovojnici

# Statistične značilnice oblike

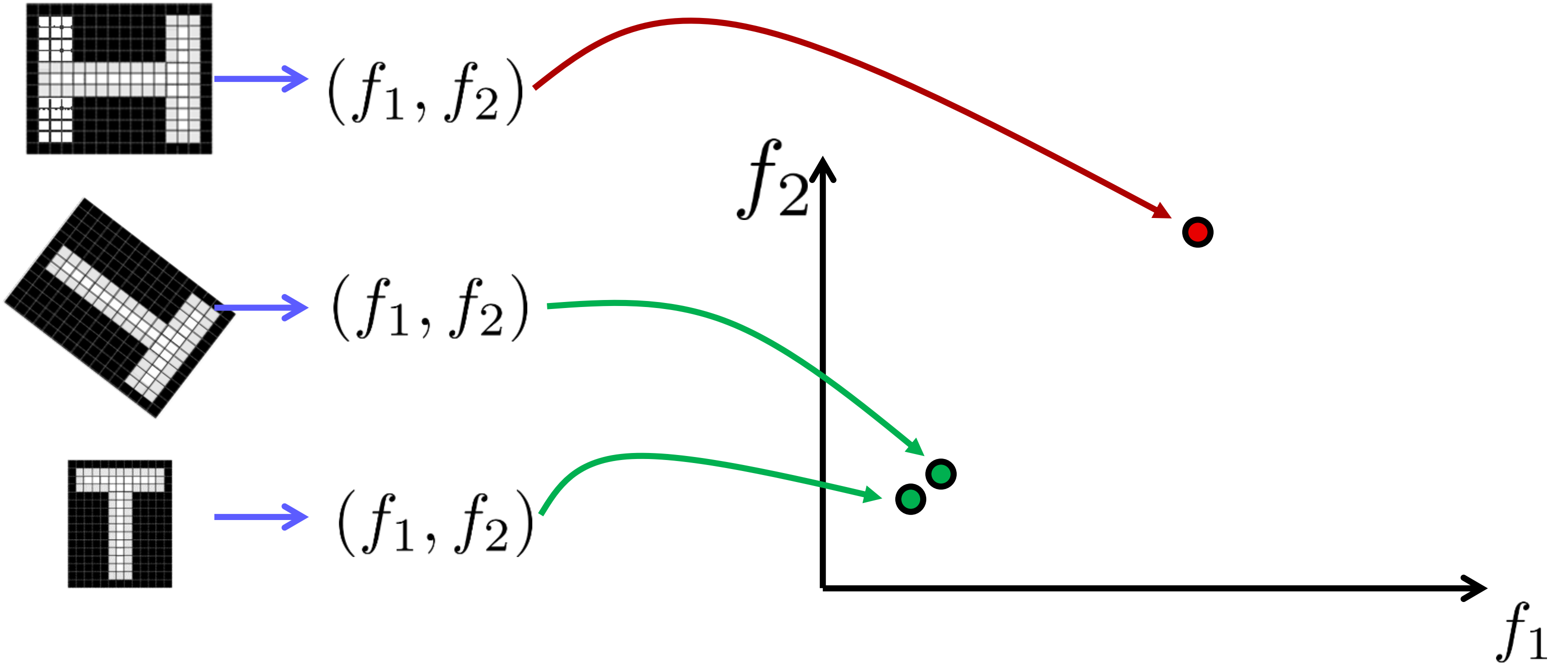
- Elemente regije obravnavamo kot točke točke porazdeljne v 2D prostoru

- Centroid
- Momenti
- Centralni momenti
- Normalizirani centralni momenti

- Geometrične lastnosti, ki temeljijo na momentih
  - Orientacija
  - Ekscentričnost
  - Invariantni momenti

# Invariante

- Želimo opisnik pod katerim sta:
  - Dve transformirani sliki istega objekta zelo podobni
  - Sliki različnih objektov zelo različni

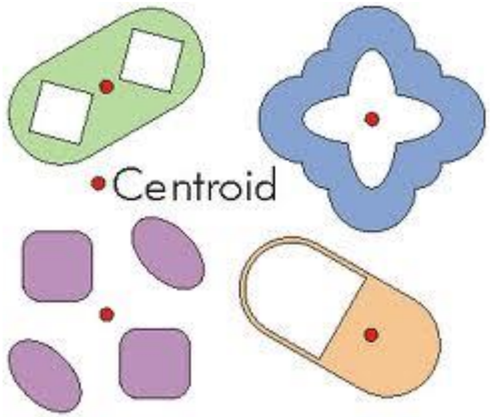


# Centroid

- Težišče regije
- Aritmetična vsota koordinat v smereh x in y

$$\bar{x} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} u \qquad \text{and} \qquad \bar{y} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} v$$

- Primeri:



# Momenti

- Splošni statistični koncept:

$$m_{pq} = \sum_{(u,v)\in\mathcal{R}} I(u,v) \cdot u^p v^q$$

- Za binarne slike:

$$m_{pq} = \sum_{(u,v)\in\mathcal{R}} u^p v^q$$

- Površina:

$$A(\mathcal{R}) = |\mathcal{R}| = \sum_{(u,v)\in\mathcal{R}} 1 = \sum_{(u,v)\in\mathcal{R}} u^0 v^0 = m_{00}(\mathcal{R})$$

- Centroid:

$$\bar{x} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} u^1 v^0 = \frac{m_{10}(\mathcal{R})}{m_{00}(\mathcal{R})} \qquad \bar{y} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} u^0 v^1 = \frac{m_{01}(\mathcal{R})}{m_{00}(\mathcal{R})}$$

# Centralni momenti

- Vzamemo centroid za referenčno točko – središče koordinatnega sistema

$$\mu_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} I(u,v) \cdot (u-\bar{x})^p \cdot (v-\bar{y})^q$$

- Za binarne slike (regije):

$$\mu_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} (u-\bar{x})^p \cdot (v-\bar{y})^q$$

- Momenti tako niso več odvisni od položaja regije na sliki

- Normalizirani centralni momenti:
  - Normalizirati moramo za faktor $s^{(p+q+2)}$

$$\bar{\mu}_{pq}(\mathcal{R}) = \mu_{pq} \cdot \left(\frac{1}{\mu_{00}(\mathcal{R})}\right)^{(p+q+2)/2}$$

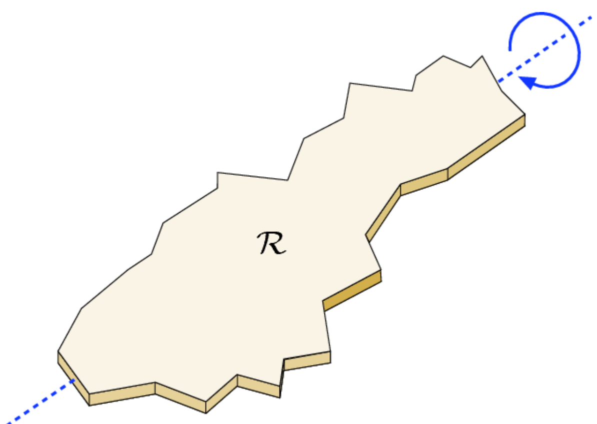
  - Centralni momenti so tako invariantni tudi na skalo
  - Primerni za klasifikacijo regij

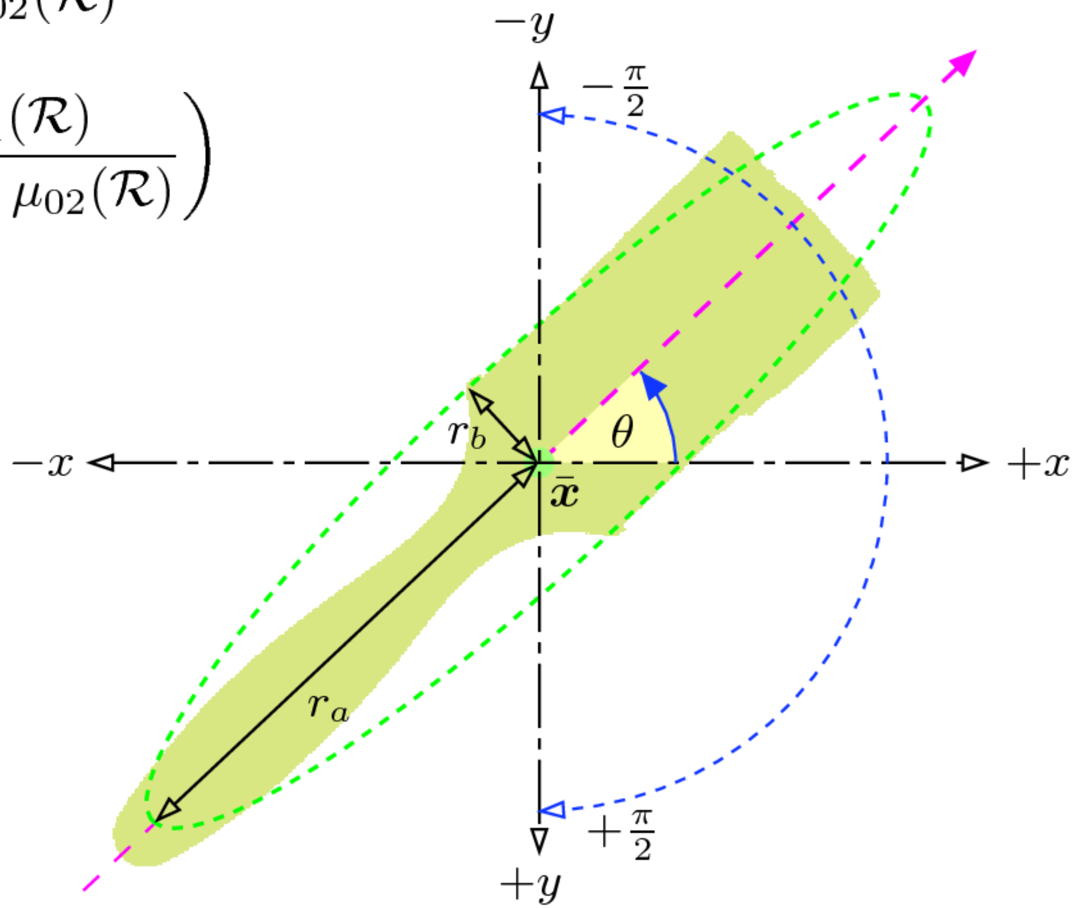
# Orientacija

- Smer glavne osi

$$\tan(2\,\theta_{\mathcal{R}}) = \frac{2 \cdot \mu_{11}(\mathcal{R})}{\mu_{20}(\mathcal{R}) - \mu_{02}(\mathcal{R})}$$

$$\theta_{\mathcal{R}} = \frac{1}{2}\tan^{-1}\left(\frac{2 \cdot \mu_{11}(\mathcal{R})}{\mu_{20}(\mathcal{R}) - \mu_{02}(\mathcal{R})}\right)$$



# Orientacija

- Vizualizacija vektorja smeri orientacije:

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \lambda \cdot \boldsymbol{x}_d = \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \lambda \cdot \begin{pmatrix} \cos(\theta_{\mathcal{R}}) \\ \sin(\theta_{\mathcal{R}}) \end{pmatrix}$$

$$\tan(2\theta_{\mathcal{R}}) = \frac{2 \cdot \mu_{11}(\mathcal{R})}{\mu_{20}(\mathcal{R}) - \mu_{02}(\mathcal{R})} = \frac{A}{B} = \frac{\sin(2\theta_{\mathcal{R}})}{\cos(2\theta_{\mathcal{R}})}$$

$$x_d = \cos(\theta_{\mathcal{R}}) = \begin{cases} 0 & \text{for} \quad A = B = 0 \\ \left[\frac{1}{2}\left(1 + \frac{B}{\sqrt{A^2+B^2}}\right)\right]^{\frac{1}{2}} & \text{otherwise,} \end{cases}$$

$$y_d = \sin(\theta_{\mathcal{R}}) = \begin{cases} 0 & \text{for} \quad A = B = 0 \\ \left[\frac{1}{2}\left(1 - \frac{B}{\sqrt{A^2+B^2}}\right)\right]^{\frac{1}{2}} & \text{for} \quad A \geq 0 \\ -\left[\frac{1}{2}\left(1 - \frac{b}{\sqrt{A^2+B^2}}\right)\right]^{\frac{1}{2}} & \text{for} \quad A < 0, \end{cases}$$

# Ekscentričnost

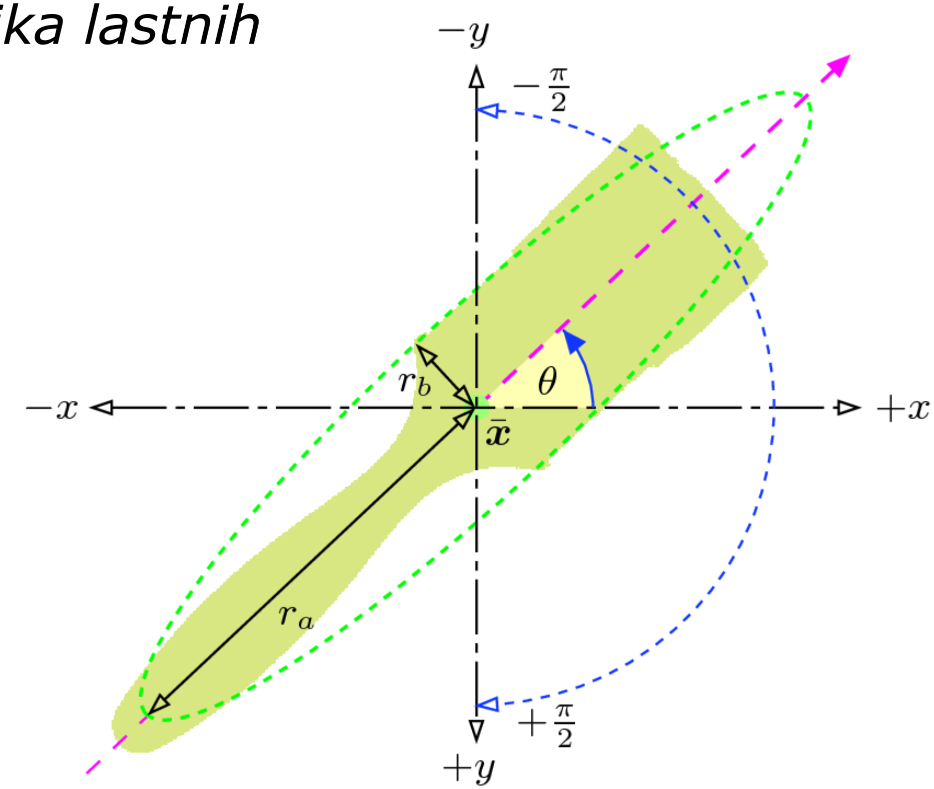
- Poldogovatost regije

$$\mathrm{Ecc}(\mathcal{R}) = \frac{a_1}{a_2} = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4 \cdot \mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4 \cdot \mu_{11}^2}}$$

- *$a_1 = 2\lambda_1$, $a_2 = 2\lambda_2$ sta večkratnika lastnih vrednosti matrike*

$$\boldsymbol{A} = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

- Vrednosti med 1 in ∞
  - Ecc=1 => krog
  - Ecc>1 => podolgovata regija
- Invariantna na orientacijo in velikost



# Ekscentričnost

- Vizualizacija elipse, ki ponazarja podolgovatost
- Dolžine osi elipse:

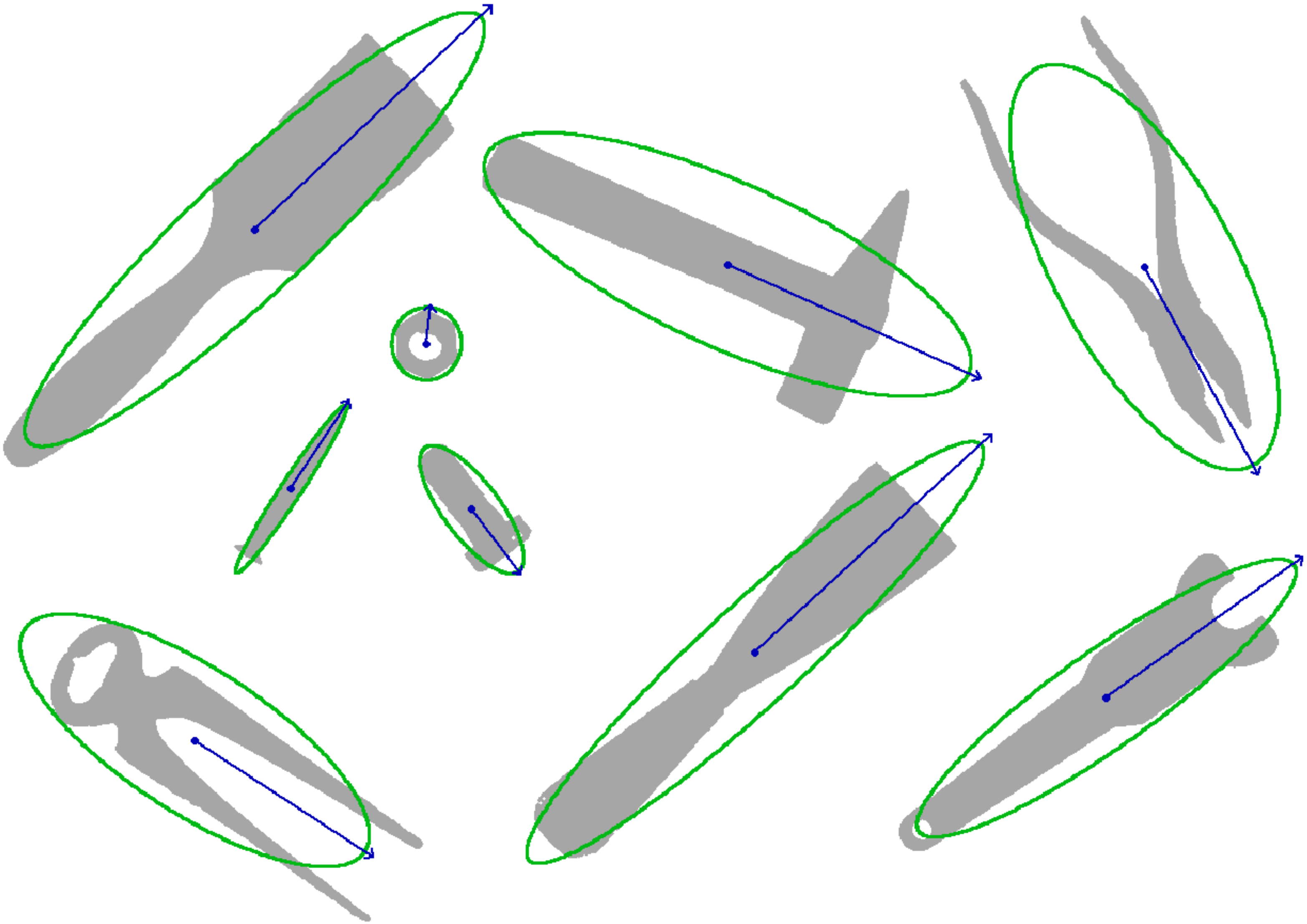
$$r_a = 2 \cdot \left(\frac{\lambda_1}{|\mathcal{R}|}\right)^{\frac{1}{2}} = \left(\frac{2\,a_1}{|\mathcal{R}|}\right)^{\frac{1}{2}}$$

$$r_b = 2 \cdot \left(\frac{\lambda_2}{|\mathcal{R}|}\right)^{\frac{1}{2}} = \left(\frac{2\,a_2}{|\mathcal{R}|}\right)^{\frac{1}{2}}$$

- Parametrična enačba elipse:

$$\begin{aligned}\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} &= \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} r_a \cdot \cos(t) \\ r_b \cdot \sin(t) \end{pmatrix} \\ &= \begin{pmatrix} \bar{x} + \cos(\theta) \cdot r_a \cdot \cos(t) - \sin(\theta) \cdot r_b \cdot \sin(t) \\ \bar{y} + \sin(\theta) \cdot r_a \cdot \cos(t) + \cos(\theta) \cdot r_b \cdot \sin(t) \end{pmatrix}\end{aligned}$$

# Primer

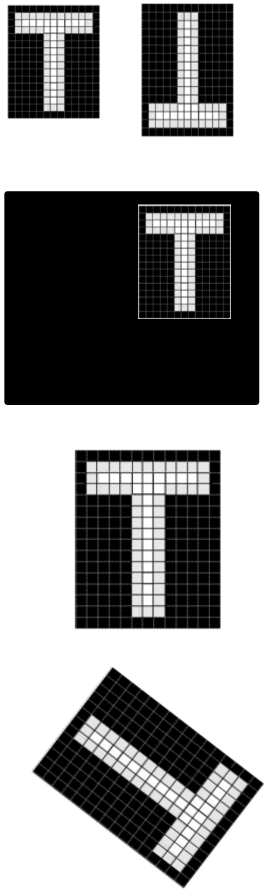


# Invariantni momenti

- Hujevi momenti oz. Invariantni momenti
- So invariantni tudi na orientacijo

Invariantni na spremembe v merilu, translaciji, zrcaljenju in rotaciji.

$$
\begin{aligned}
H_1 &= \bar{\mu}_{20} + \bar{\mu}_{02} \\
H_2 &= (\bar{\mu}_{20} - \bar{\mu}_{02})^2 + 4\,\bar{\mu}_{11}^2 \\
H_3 &= (\bar{\mu}_{30} - 3\,\bar{\mu}_{12})^2 + (3\,\bar{\mu}_{21} - \bar{\mu}_{03})^2 \\
H_4 &= (\bar{\mu}_{30} + \bar{\mu}_{12})^2 + (\bar{\mu}_{21} + \bar{\mu}_{03})^2 \\
H_5 &= (\bar{\mu}_{30} - 3\,\bar{\mu}_{12}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + (3\,\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot \left[3\,(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
H_6 &= (\bar{\mu}_{20} - \bar{\mu}_{02}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + 4\,\bar{\mu}_{11} \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \\
H_7 &= (3\,\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3\,(\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + (3\,\bar{\mu}_{12} - \bar{\mu}_{30}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot \left[3\,(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right]
\end{aligned}
$$



- Običajno se uporablja logaritme vrednosti

# Projekcije

- Enodimenzionalna predstavitev vsebine slike
- Slikovne elemente projiciramo na x in y koordinato:

$$P_{\text{hor}}(v_0) = \sum_{u=0}^{M-1} I(u, v_0) \qquad \text{for} \quad 0 < v_0 < N$$

$$P_{\text{ver}}(u_0) = \sum_{v=0}^{N-1} I(u_0, v) \qquad \text{for} \quad 0 < u_0 < M$$

- Včasih tako lahko ločimo dokument na smiselne celote
- Projiciramio lahko tudi na glavni osi regije

# Topološke lastnosti

- Zajamejo strukturne lastnosti
- So invariantne tudi na zelo močne transformacije slike

- Konveksnost regije

- Število lukenj: $N_L(\mathcal{R})$

- Eulerjevo število: $N_E(\mathcal{R}) = N_R(\mathcal{R}) - N_L(\mathcal{R})$