Robotika in računalniško zaznavanje (RRZ)

Filtriranje slik

Danijel Skočaj

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Literatura: W. Burger, M. J. Burge (2008).
Digital Image Processing, poglavje 6

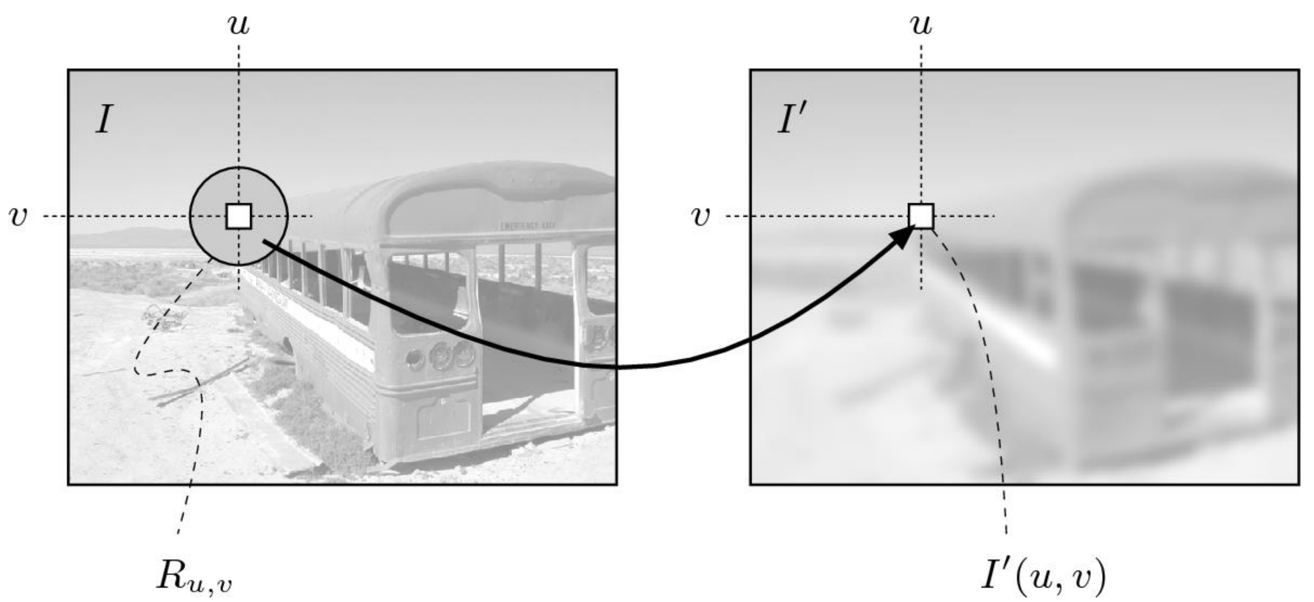
v7.0

Filtriranje

- Filtriranje je operacija, ki
  - Ne spremeni geometrije slike
  - Uporabi več kot en slikovni element za izračun vrednosti novega slikovnega elementa

Filtriranje

- Za izračun $I'(u,v)$ uporabi celotno regijo $R_{u,v}$ s slike I



Povprečni filter

- Primer: povprečni filter
  - Vrednost slikovnega elementa naj bo enaka srednji vrednosti elementov v okolici

$$I'(u,v) \leftarrow \frac{p_0+p_1+p_2+p_3+p_4+p_5+p_6+p_7+p_8}{9}$$

$$\begin{aligned} I'(u,v) \leftarrow \tfrac{1}{9} \cdot [\, & I(u-1,v-1) && + I(u,v-1) && + I(u+1,v-1) && + \\ & I(u-1,v) && + I(u,v) && + I(u+1,v) && + \\ & I(u-1,v+1) && + I(u,v+1) && + I(u+1,v+1) \,] && \end{aligned}$$

$$I'(u,v) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i,v+j)$$

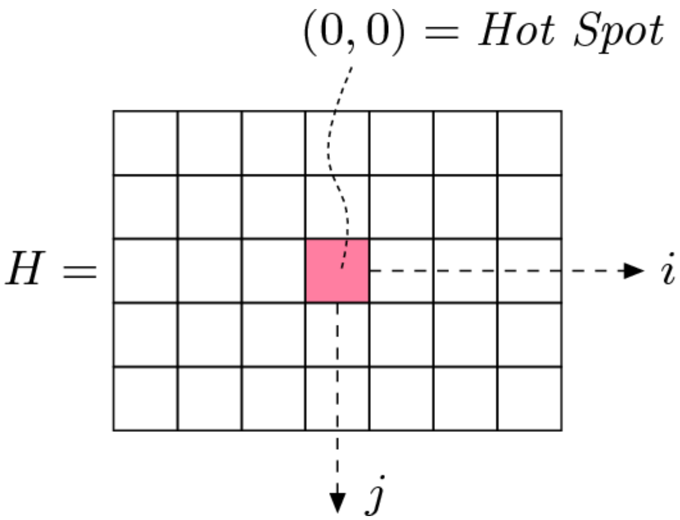
Filtri

- Različne parametri in vrste filtrov
  - Matematične lastnosti filtra
    - Linearni
    - Nelinearni
  - Velikost filtrirne regije
    - 3x3, 5x5, 7x7, 21x21, ...
  - Oblika filtrirne regije
    - Kvadratna
    - Okrogla (za izotropično filtriranje)
  - Uteži
    - Različni slikovni elementi v filtrirni regiji so različno pomembni
    - Tudi filter v obliki prstana

- Filter = sito

Linearni filtri

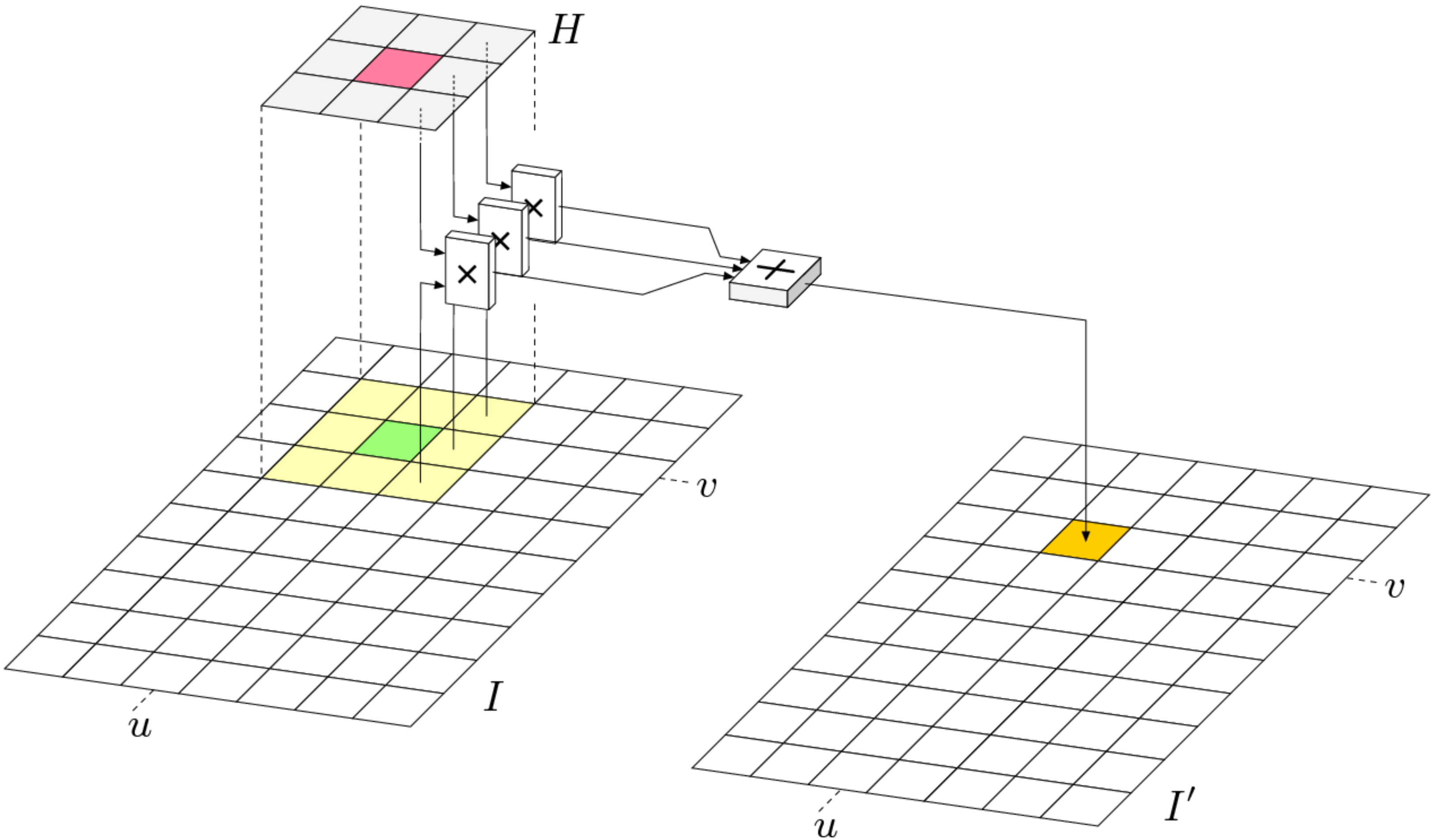
- Kombinirajo vrednosti slikovnih elementov s filtrirane regije na linearen način kot uteženo vsoto
- Filtrirna matrika (ali maska) *H(i,j)*
  - Diskretna
  - Dvodimenzionalna
  - Realna funkcija (matrika)
  - S koordinatnim sistemom (z izhodiščnem v središčnem elementu)
  - Definirana s koeficienti filtra



- Primer: povprečni filter:

$$H(i,j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Uporaba filtra



Uporaba filtra

- Tri koraki:
  - Center maske filtra poravnamo s slikovnim elementom na sliki
  - Zmnožimo vse koeficiente filtra z vrednostmi istoležnih slikovnih elementov
  - Vsoto shranimo kot novo vrednost slikovnega elementa

- Formalni zapis:

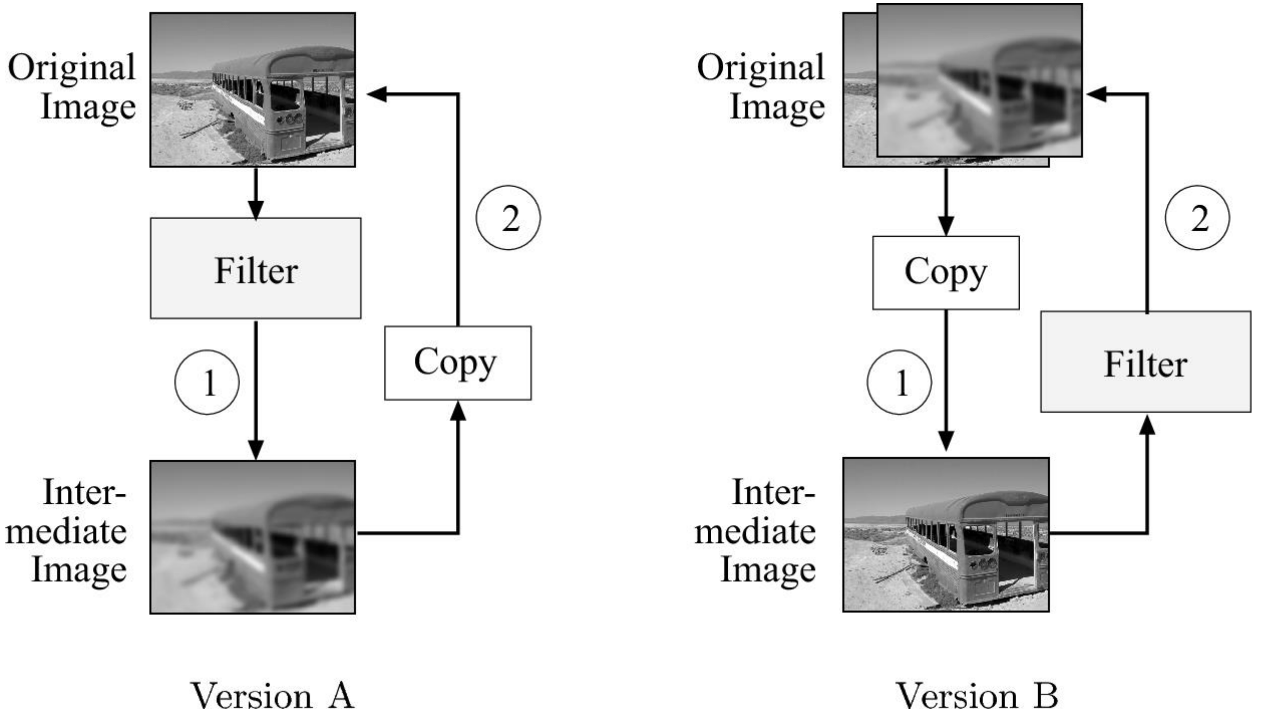
$$I'(u,v) \leftarrow \sum_{(i,j)\in R_H} I(u+i, v+j) \cdot H(i,j)$$

- Primer: maska filtra 3x3:

$$I'(u,v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u+i, v+j) \cdot H(i,j)$$

Implementacija filtra

- Za izračun vrednosti novih slikovnih elementov se uporabljajo vrednosti z originalne slike
  - Uporaba vmesne slike



Algoritem: 3x3 povprečni filter

```java
import ij.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;

public class Filter_Average3x3 implements PlugInFilter {
    ...
    public void run(ImageProcessor orig) {
        int w = orig.getWidth();
        int h = orig.getHeight();
        ImageProcessor copy = orig.duplicate();

        for (int v = 1; v <= h-2; v++) {
            for (int u = 1; u <= w-2; u++) {
                //compute filter result for position (u,v)
                int sum = 0;
                for (int j = -1; j <= 1; j++) {
                    for (int i = -1; i <= 1; i++) {
                        int p = copy.getPixel(u+i, v+j);
                        sum = sum + p;
                    }
                }
                int q = (int) Math.round(sum/9.0);
                orig.putPixel(u, v, q);
            }
        }
    }
} // end of class Filter_Average3x3
```

Filter za glajenje

- Uteži so lahko tudi različne
- Večjo utež na bolj sredinske elemente filtra:

$$H(i,j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \mathbf{0.2} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix}$$

- Vsota koeficientov je 1
  - Globalna svetlost slike se ne spremeni
  - Vrednosti slikovnih elementov ne presežejo maksimalne vrednosti

Algoritem: 3x3 filter za glajenje

```
public void run(ImageProcessor orig) {
    int w = orig.getWidth();
    int h = orig.getHeight();
    // 3 × 3 filter matrix
    double[][] filter = {
        {0.075, 0.125, 0.075},
        {0.125, 0.200, 0.125},
        {0.075, 0.125, 0.075}
    };
    ImageProcessor copy = orig.duplicate();

    for (int v = 1; v <= h-2; v++) {
        for (int u = 1; u <= w-2; u++) {
            // compute filter result for position (u,v)
            double sum = 0;
            for (int j = -1; j <= 1; j++) {
                for (int i = -1; i <= 1; i++) {
                    int p = copy.getPixel(u+i, v+j);
                    // get the corresponding filter coefficient:
                    double c = filter[j+1][i+1];
                    sum = sum + c * p;
                }
            }
            int q = (int) Math.round(sum);
            orig.putPixel(u, v, q);
        }
    }
}
```

Celoštevilski koeficienti

- Celoštevilski koeficienti in skalarni faktor

$$H(i,j) = s \cdot H'(i,j)$$

- Skalarni faktor poskrbi za normalizacijo

$$s = \frac{1}{\sum_{i,j} H'(i,j)}$$

- Primer: 3x3 filter za glajenje

$$H(i,j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \underline{0.200} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix} = \frac{1}{40} \begin{bmatrix} 3 & 5 & 3 \\ 5 & \underline{8} & 5 \\ 3 & 5 & 3 \end{bmatrix}$$

- Bolj učinkovtio in enostavno računanje

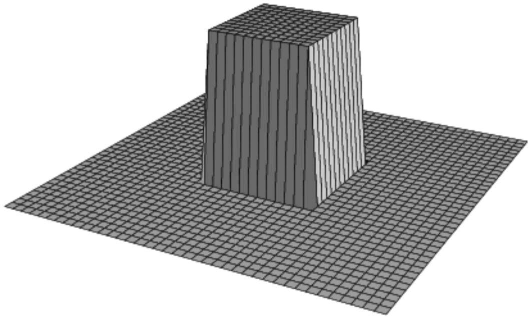
Vrste linearnih filtrov

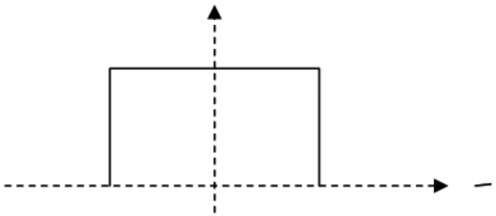
- Filtri za glajenje
  - Imajo vse koeficiente pozitivne
  - Zgladijo (zamažejo) originalno sliko
  - Zmanjšajo gaussov šum
  - Nizko-pasovno filtriranje v frekvenčni domeni
    - Odstrani vse visoke frekvence na sliki
  - Škatlast filter in gaussov filter

- Diferenčni filtri
  - Imajo tudi negativne koeficiente
  - Poudarijo lokalne spremembe v intenziteti
  - Za iskanje robov in ostrenje slike

Škatlast filter

- Najbolj enostaven filter za glajenje
- Povprečni filter
- Nima lepih lastnosti v frekvenčni domeni
  - Povzroči efekt prstanov
- Vsi koeficienti imajo enake vrednosti
- Ni izotropičen





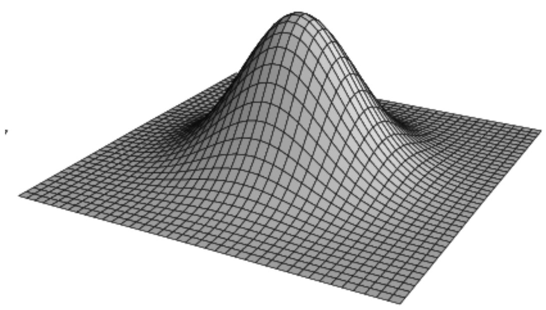
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Gausov filter

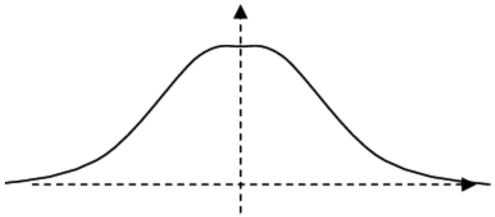
- Matrika filtra ustreza diskretni dvo-dimenzionalni Gausovi funkciji:

$$G_\sigma(r) = e^{-\frac{r^2}{2\sigma^2}}$$

$$G_\sigma(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$



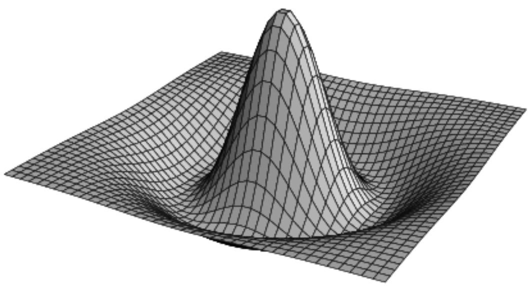
- Najbolj pomemben sredinski element
- Je izotropičen
- Ima lepe lastnosti v frekvenčnem prostoru
- Je separabilen
  - Omogoča učinkovito računanje



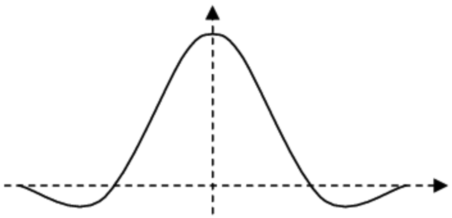
0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

Diferenčni filter

- Ima tudi negativne uteži
- Razlika dveh uteženih vsot
  - Pozitivnih koeficientov
  - Negativnih koeficientov



$$I'(u,v) = \sum_{(i,j)\in R_H^+} I(u+i, v+j) \cdot |H(i,j)| - \sum_{(i,j)\in R_H^-} I(u+i, v+j) \cdot |H(i,j)|$$



- Laplaceov filter
- Poudarja lokalne spremembe v intenziteti

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

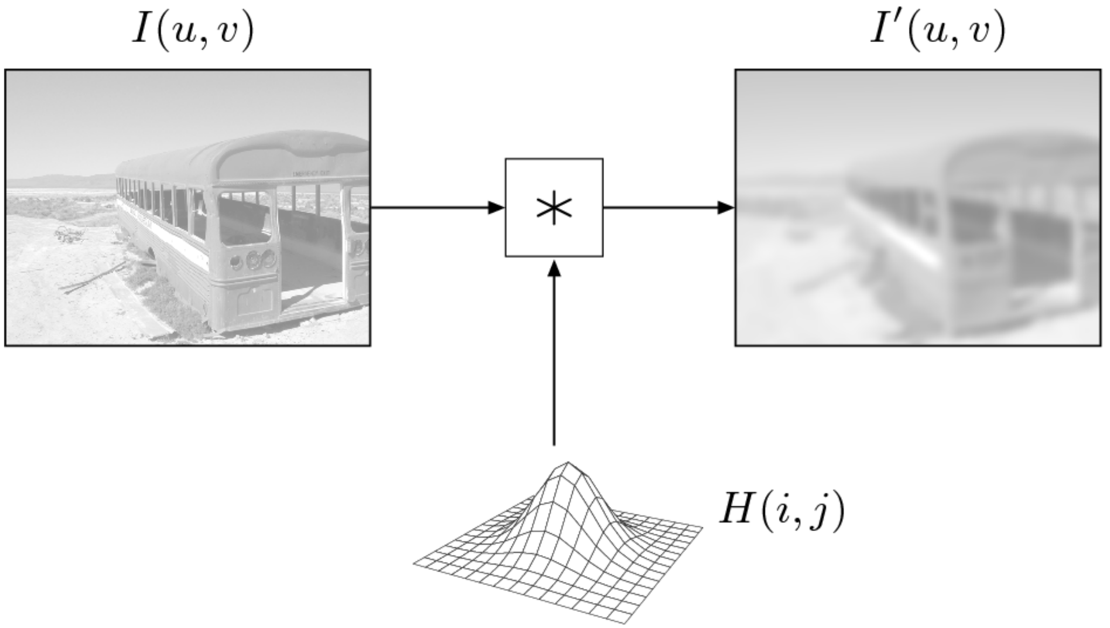
Linearna konvolucija

- Operacija linearnega filtriranje je zelo povezano z matematično linearno konvolucijo

$$I'(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i, v-j) \cdot H(i,j)$$

$$I' = I * H$$

- H je konvolucijsko jedro



Filtriranje in konvolucija

- Konvolucija je v bistvu operacija filtriranja z (horizontalno in vertikalno) zrcaljeno matriko filtra:

$$
\begin{aligned}
I'(u,v) &= \sum_{(i,j)\in R_H} I(u-i, v-j) \cdot H(i,j) \\
&= \sum_{(i,j)\in R_H} I(u+i, v+j) \cdot H(-i,-j) \\
&= \sum_{(i,j)\in R_H} I(u+i, v+j) \cdot H^*(i,j).
\end{aligned}
$$

$$H^*(i,j) = H(-i,-j)$$

- Korelacija je konvolucija z zrcaljeno matriko filtra
- Pri simetričnih jedrih sta obe matriki enaki

Lastnosti linearne konvolucije

- Komutativnost

$$I * H = H * I$$

- Linearnost

$$(s \cdot I) * H \;=\; I * (s \cdot H) \;=\; s \cdot (I * H)$$

$$(I_1 + I_2) * H \;=\; (I_1 * H) + (I_2 * H)$$

$$(b + I) * H \;\neq\; b + (I * H)$$

- Asociativnost

$$A * (B * C) = (A * B) * C$$

Ločljivost linearnih filtrov

- Konvolucijsko jedro je ločljivo, če ga lahko izrazimo kot konvolucijo večih jeder

$$H = H_1 * H_2 * \ldots * H_n$$

- Potem lahko tudi konvolucijo opravimo kot zaporedje večjega števila konvolucij (z manjšimi jedri)

$$\begin{aligned} I * H &= I * (H_1 * H_2 * \ldots * H_n) \\ &= \big(\ldots((I * H_1) * H_2) * \ldots * H_n\big) \end{aligned}$$

- Z uporabo ločljivih jeder lahko velikokrat zelo pohitrimo računanje

x/y ločljivost

- Ločitev 2-D jedra *H* v dve 1-D jedri H_x in H_y

- Primer:

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

$$I' \leftarrow (I * H_x) * H_y = I * \underbrace{(H_x * H_y)}_{H_{xy}}$$

$$H_{xy} = H_x * H_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

  - 2D filter: 5x3=15 operacij
  - 2x1D filtra: 5+3=8 operacij

Pohitritev filtriranja

- Uporaba ločjivih filtrov lahko zelo pohitri filtriranje
- V splošnem:
  - Velikost slike: *MxN*
  - Velikost jedra filtra: *(2K+1)x(2L+1)*
  - Neposredna implementacija:
    $2K\ 2L\ M\ N = 4KLMN$ operacij
  - Če je jedro ločljivo na dva 1D filtra velikosti *(2K+1)* in *(2L+1)*
  - Učinkvita implementacija:
    $((2K+1) + (2L+1))\ M\ N = 2MN(K+L+1)$

- Računska kompleksnost (če sta slika in filter velikosti NxN)
  - Neposredna implementacija: $O(N^4)$
    - Kvadratna rast z velikostjo (stranice) filtra
  - Učinkovita implementacija: $O(N^3)$
    - Linearna rast z velikostjo (stranice) filtra

Ločljivi Gaussovi filtri

- Pogoj za x/y ločljivost 2D linearnega filtra:

$$H_{x,y}(i,j) = (H_x \otimes H_y)(i,j) = H_x(i) \cdot H_y(j)$$

- Gaussovo funkcijo lahko lepo razbijemo:

$$G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}} = g_\sigma(x) \cdot g_\sigma(y)$$

- 2D Gaussov filter lahko tako implementiramo kot dva 1D Gaussova filtra:

$$I' \leftarrow I * H^{G,\sigma} = I * H_x^{G,\sigma} * H_y^{G,\sigma}$$

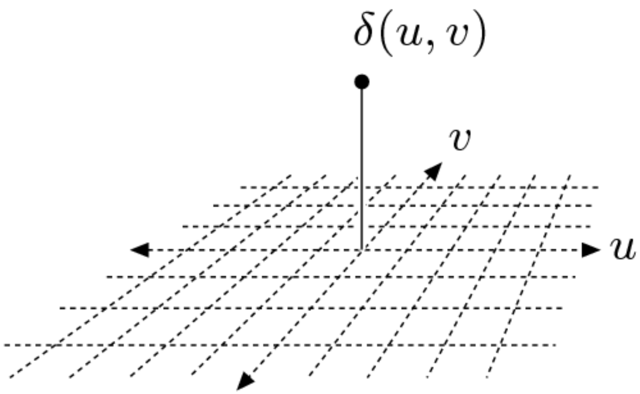
- Jedro naj bo veliko pribl. $\pm 2.5\sigma$ do $\pm 3.5\sigma$
  - Da se izognemo zaokrožitvenim učinkom
  - Primer: σ=10 zahteva filter velikosti 51x51

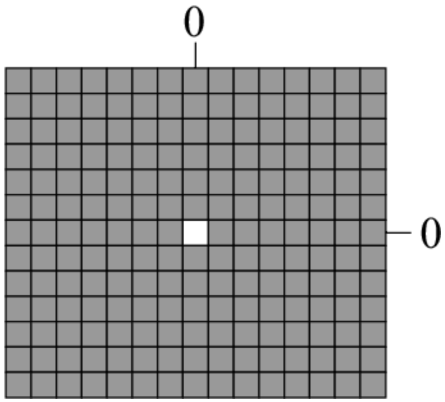
Impulz oz. Diracova funkcija

- Impulzna oz. Diracova funkcija:

$$I * \delta = I$$

$$\delta(u,v) = \begin{cases} 1 & \text{for } u = v = 0 \\ 0 & \text{otherwise.} \end{cases}$$





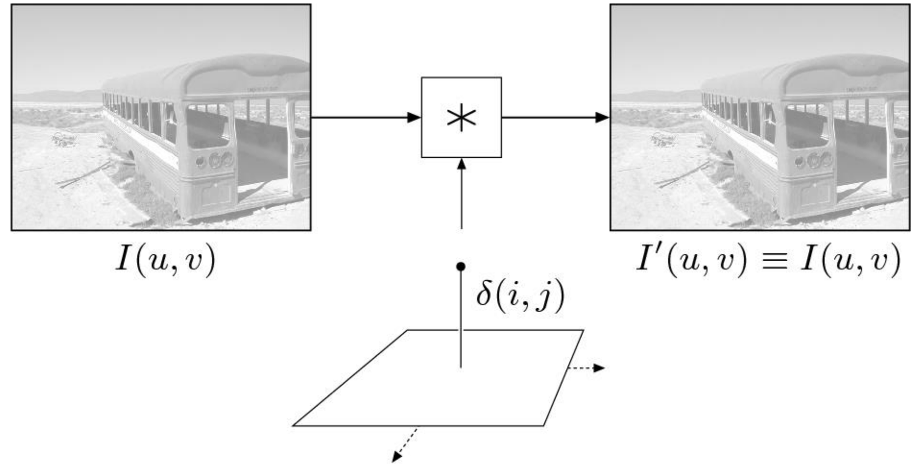
Impulzni odziv filtra

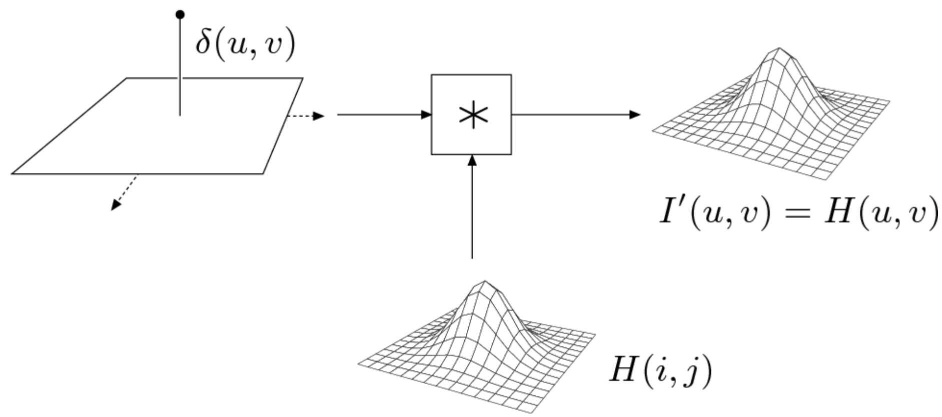
- Filtriranje z Diracovo funkcijo ne spremeni slike
- Rezultat filtiranja slike z Diracovo funkcijo je filter

$$H * \delta = \delta * H = H$$

-> Impulzni odziv filtra

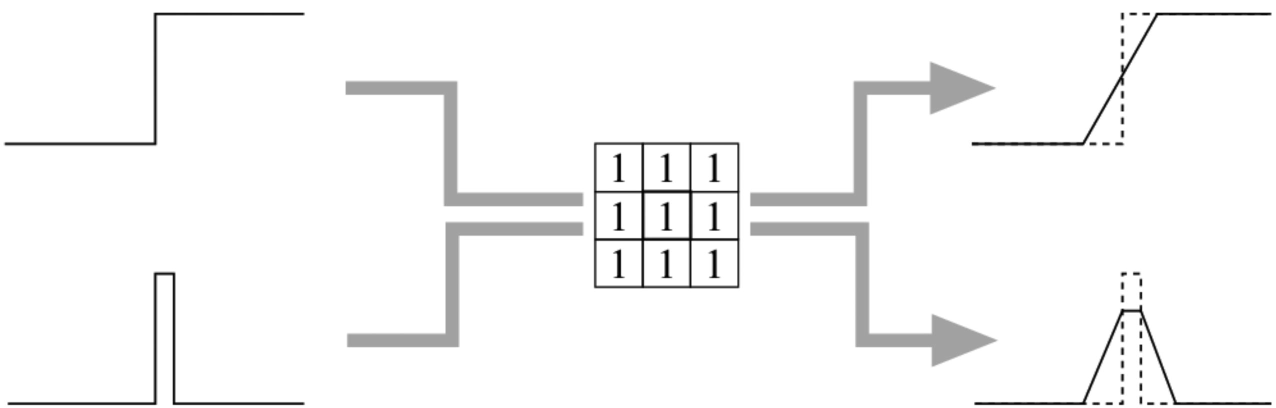
- Nam pove kakšen je filter





Pomanjkljivost linearninh filtrov

- Linearni filtri imajo veliko pomanjkljivost: gladijo slike (tudi robove, črte,in ostale strukture na sliki)
  - In to ne selektivno, po celotni sliki enako



Nelinearni filtri

- Nelinearni filtri tudi računajo vrednost slikovnega elementa s pomočjo vrednosti slikovnih elementov z določene regije na originalni sliki
  - Uporabljajo pa nelinearno funkcijo!
- Nelinearni filtri nimajo lepih lastnosti linearnih filtrov
- Se pa tudi izognejo slabim lastnostim (uniformnemu glajenju)

- Razlikujejo se glede na nelinearno funkcijo, ki jo implementirajo:
  - Minimalni in maksimalni filter
  - Madianin filter
  - Uteženi medianin filter
  - Ostali filtri

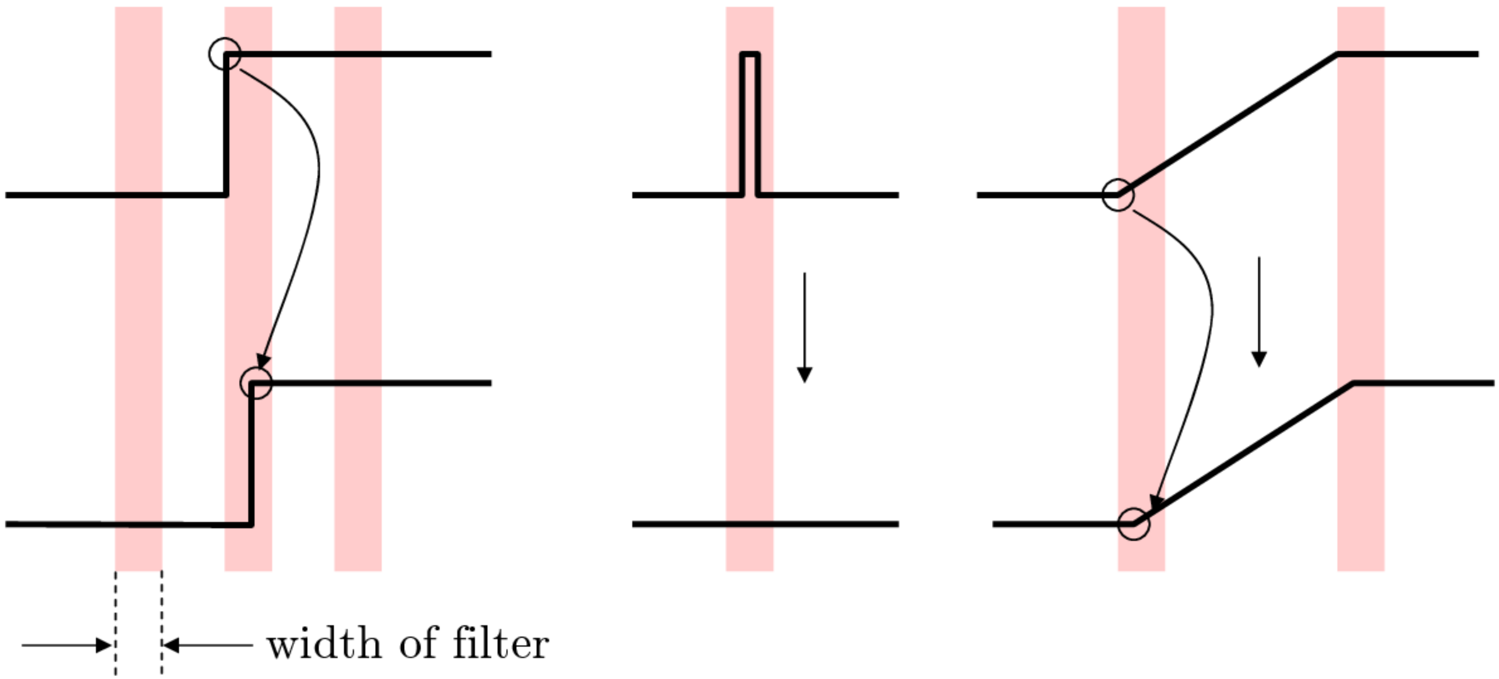
Minimalni in maksimalni filter

- Slikovnemu elementu dodelimo min. oz. maksimalno vrednost regije:

$$I'(u,v) \leftarrow \min\{I(u+i,v+j) \mid (i,j) \in R\}$$

$$I'(u,v) \leftarrow \max\{I(u+i,v+j) \mid (i,j) \in R\}$$

- Vpliv enodimenzionalnega minimalnega filtra:



Učinek min. in maks. filtrov

- Impulzni šum (sol in poper)
  - 3x3 min. in maks. filter
  - Minimalni filter odstrani bele pike in razširi temne regije
  - Maksimalni filter odstrani črne pike in razširi svetle regije

Medianin filter

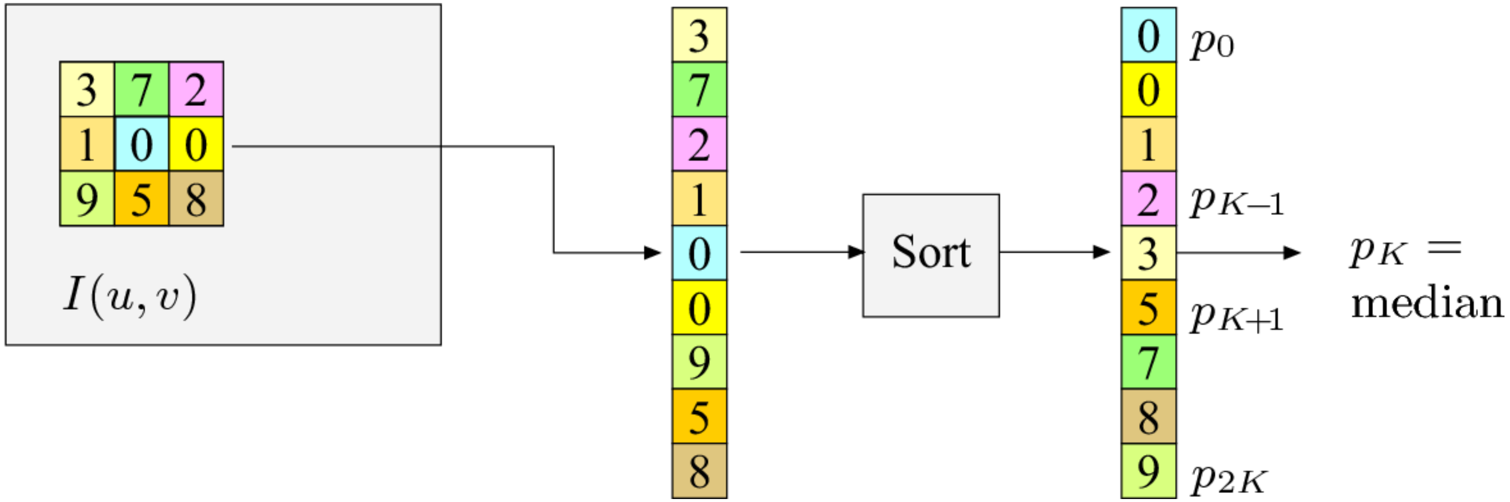
- Slikovnemu elementu dodelimo vrednost mediane okoliške regije

$$I'(u,v) \leftarrow \text{median}\,\{I(u+i,v+j) \mid (i,j) \in R\}$$

  - kjer je mediana definirana kot

$$\text{median}\,(p_0, p_1, \ldots, p_K, \ldots, p_{2K}) \triangleq p_K$$
$$\text{median}\,(p_0, \ldots, p_{K-1}, p_K, \ldots, p_{2K-1}) \triangleq (p_{K-1} + p_K)\,/\,2$$

- Mediana je robustna statistika
  - Posamezni odstopajoči elementi nimajo velikega vpliva
- Primer:

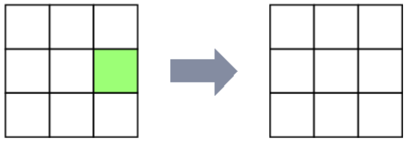


Algoritem

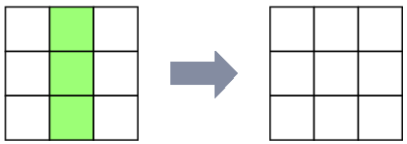
```
public class Filter_Median3x3 implements PlugInFilter {
    final int K = 4; // filter size

    public void run(ImageProcessor orig) {
        int w = orig.getWidth();
        int h = orig.getHeight();
        ImageProcessor copy = orig.duplicate();

        // vector to hold pixels from 3×3 neighborhood
        int[] P = new int[2*K+1];

        for (int v = 1; v <= h-2; v++) {
            for (int u = 1; u <= w-2; u++) {
                // fill the pixel vector P for filter position u, v
                int k = 0;
                for (int j = -1; j <= 1; j++) {
                    for (int i = -1; i <= 1; i++) {
                        P[k] = copy.getPixel(u+i, v+j);
                        k++;
                    }
                }
                // sort pixel vector and take the center element
                Arrays.sort(P);
                orig.putPixel(u, v, P[K]);
            }
        }
    }

} // end of class Filter_Median3x3
```

Učinek medianinega filtra

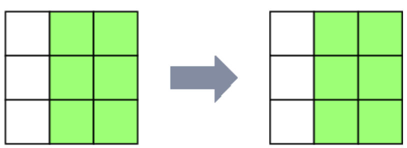
- Izolirane točke se odstranijo



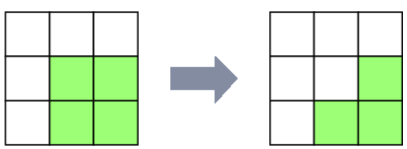
- Tanke črte se odstranijo



- Robovi se ohranijo



- Vogali se zaokoržijo



Učinek medianinega filtra

- Odstranjevanje impulznega šuma

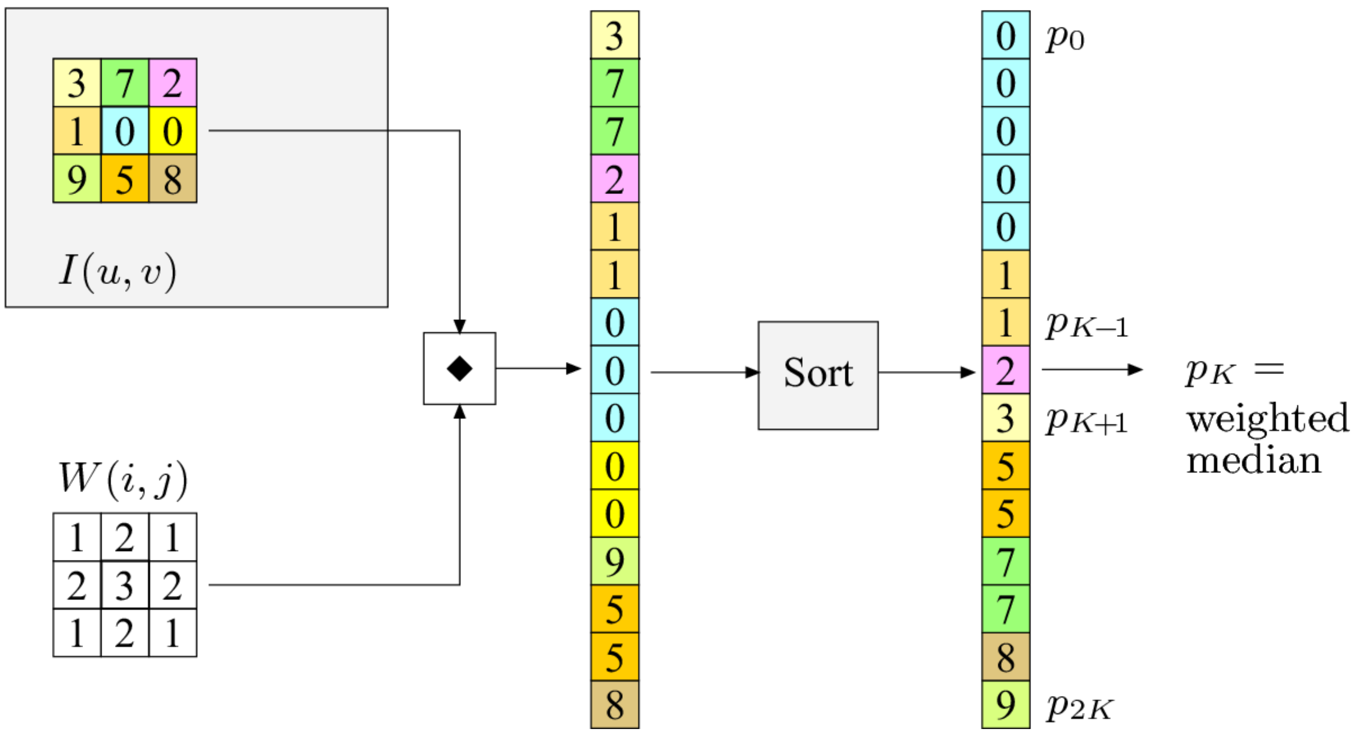
originalna slika | 3x3 škatlast linearen filter | 3x3 medianin filter

Utežen medianin filter

- Posameznim elementom v filtru dodamo uteži – število glasov za posamezno vrednost
  - Vsaka vrednost slikovnega elementa se (lahko) pojavi večkrat v vektorju z vsemi vrednostmi slikovnih elementov v regiji:

$$Q = (p_0, \ldots, p_{L-1}) \quad \text{of length} \quad L = \sum_{(i,j)\in R} W(i,j)$$

- Primer:



Utežen medianin filter

- Primer, ki bolj uteži sredinski slikovni element od okolice:

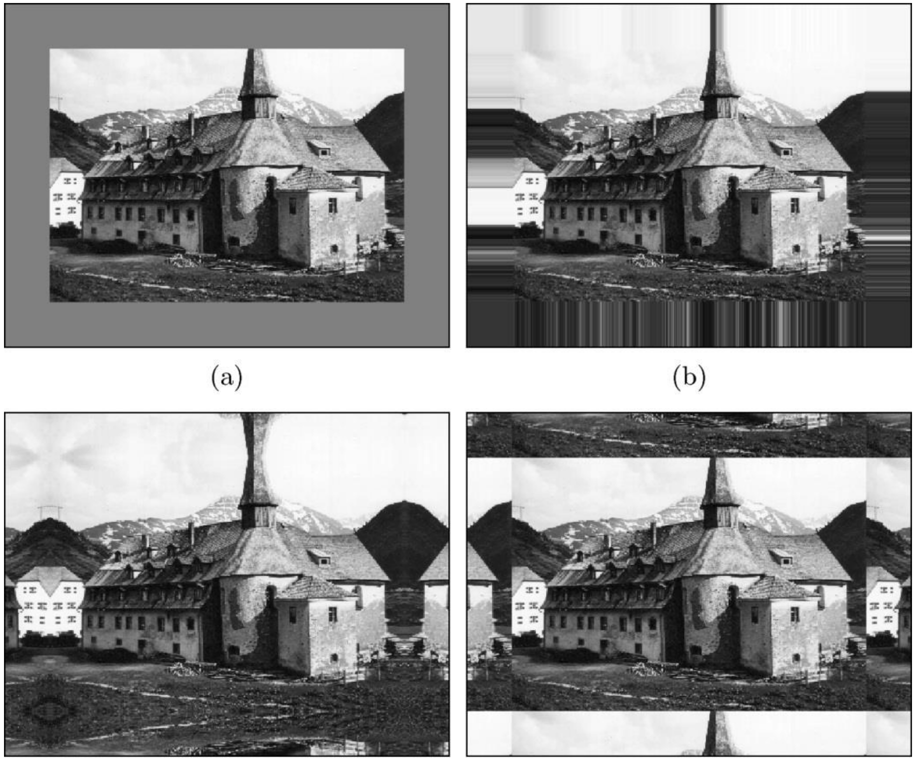
$$W(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & \mathbf{3} & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Na ta način lahko realiziramo tudi nepravokotne medianine filtre
  - Primer křižnega medianinega filtra:

$$W^{+}(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Kaj z robovi slik?

- Slikovne elemente na robu slike ne moremo sprocesirati
- Več ad-hoc rešitev:
  - Vrednosti teh slikovnih elementov postavimo na neko vrednost
  - Vrednosti teh slikovnih elementov ne spremenimo
  - Razširimo sliko, da lahko računamo tudi na robu:



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