

# OpenCV

*Andrej Lúčny*

*Katedra aplikovanej informatiky FMFI UK*

*lucny@fmph.uniba.sk*

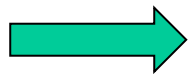
*[http://dai.fmph.uniba.sk/w/Andrej\\_Lucny](http://dai.fmph.uniba.sk/w/Andrej_Lucny)*

*www.agentspace.org/opencv*

3

# Zmena veľkosti

Pozor na artefakty!



Spôsob  
interpolácie

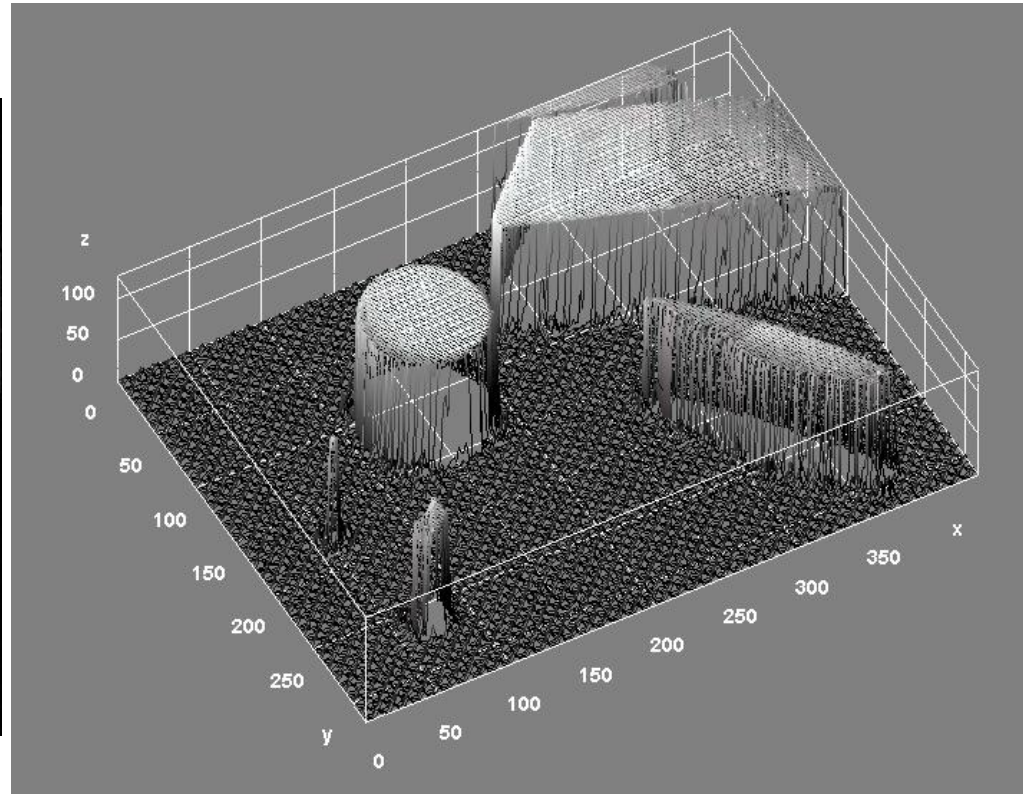
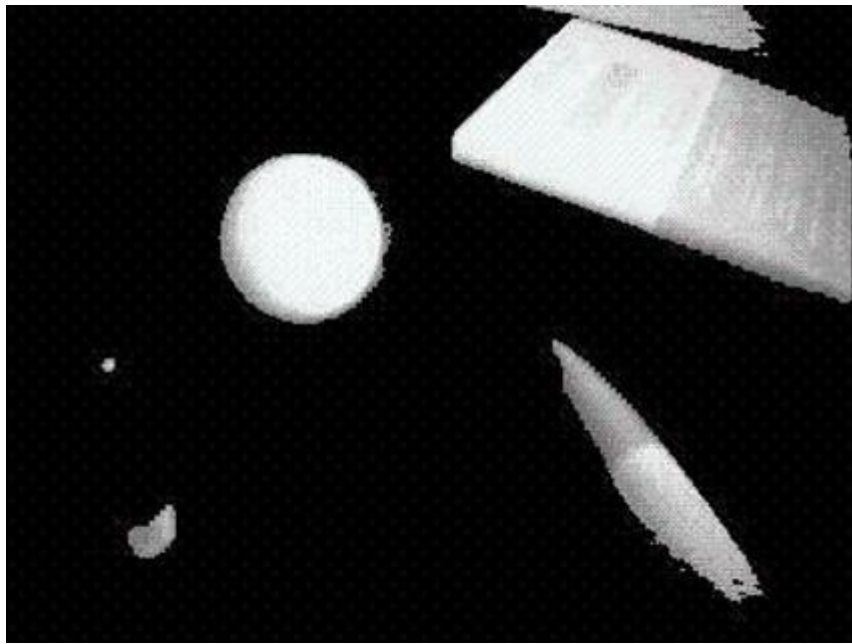


```
larger = cv2.resize(normal, dims, cv2.INTER_CUBIC)
```



```
smaller = cv2.resize(normal, dims, cv2.INTER_AREA)
```

# Obraz ako funkcia dvoch premenných

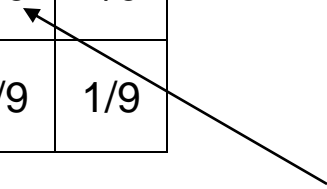


# Operátory (kernely)

Operátor = mení funkciu na funkciu (napríklad derivácia)

Operátor môžeme definovať pomocou kernelu

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Kernel sa prikladá stredom na pixel a počíta sa podľa neho vážený priemer  
problém: okraje

# Blur

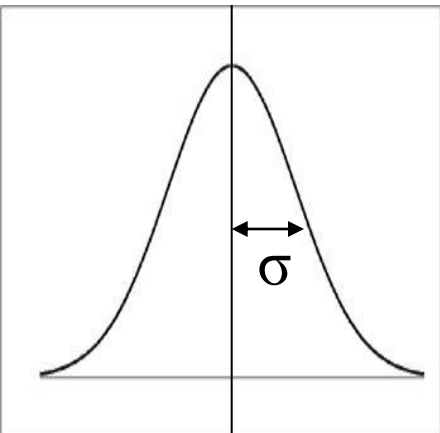
blur kernel 3x3

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



```
frame = cv2.blur(frame, (3, 3))
```

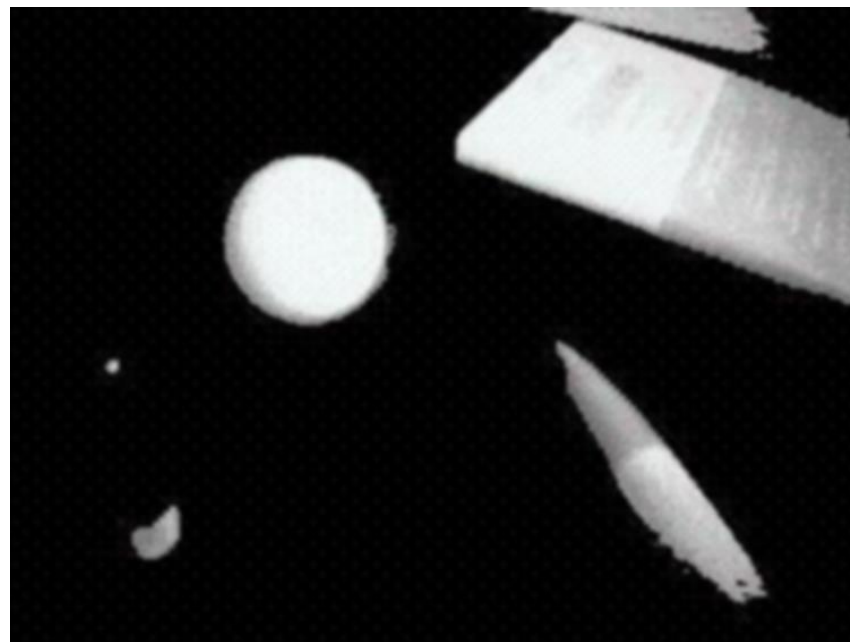
Pixel je nahradený priemerom okolia, čo potláča šum



# Gaussian blur

blur kernel 3x3

0.06	0.13	0.06
0.13	0.25	0.13
0.06	0.13	0.06



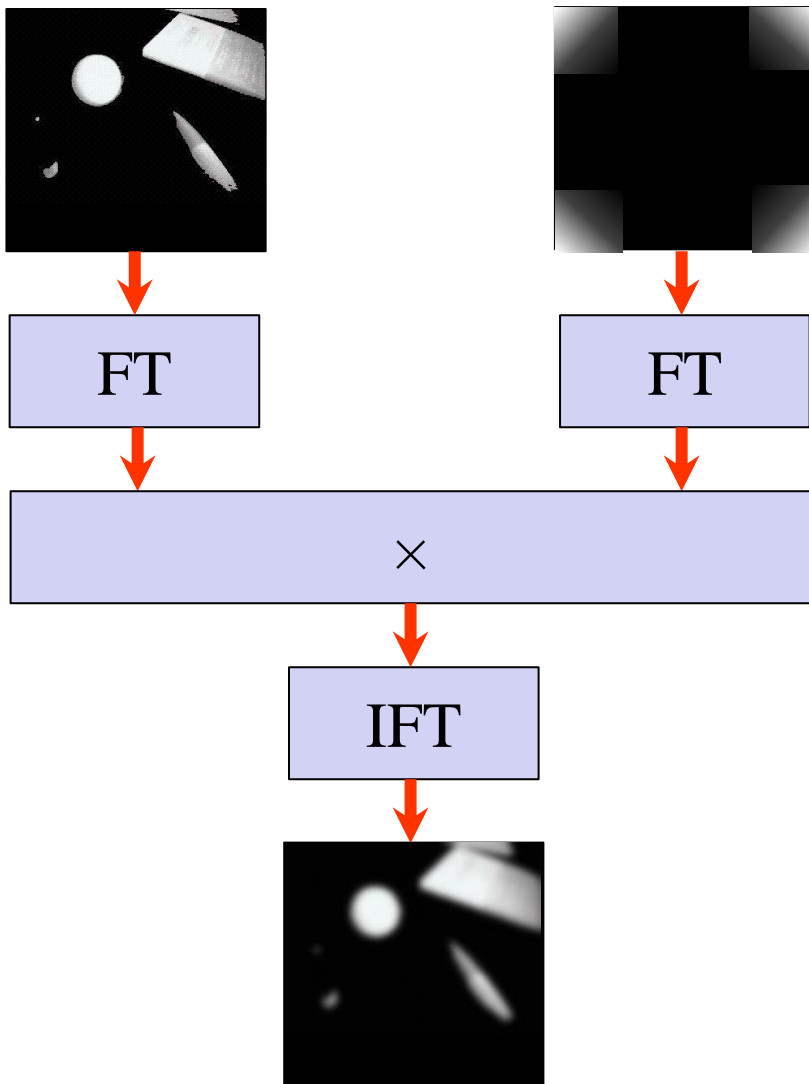
```
frame = cv2.GaussianBlur(frame, (5, 5), 0)
```

*ksize*

*σ = 0 ... počíta sa podľa ksize*

# Large window Gaussian Blur

Keď je okolie veľké, filtre sa počítajú vo Furierovom spektre, takže miesto  $O(pq)$  (počet pixelov obrázku krát počet pixelov okolia) to je  $O(p \log(p))$ , čo je pri veľkom okolí oveľa rýchlejšie



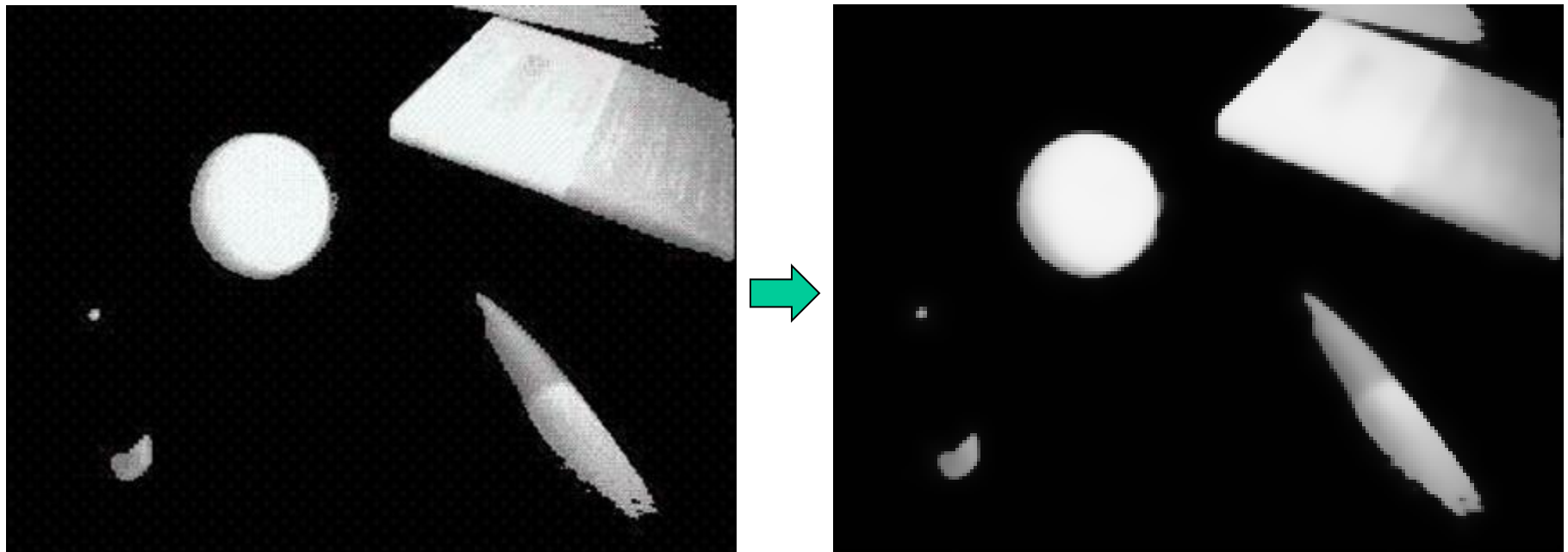
```
kernel1D = cv2.getGaussianKernel(ksize,0)
kernel2D = kernel1D * np.transpose(kernel1D)
frame = cv2.filter2D(frame,kernel2D)
```

# Bilaterálny filter

15 80 80

```
blured = cv2.bilateralFilter(gray, diameter, sigmaColor, sigmaSpace)
```

čím je okolitá hodnota bližšie a čím je podobnejšia tým viac vplýva

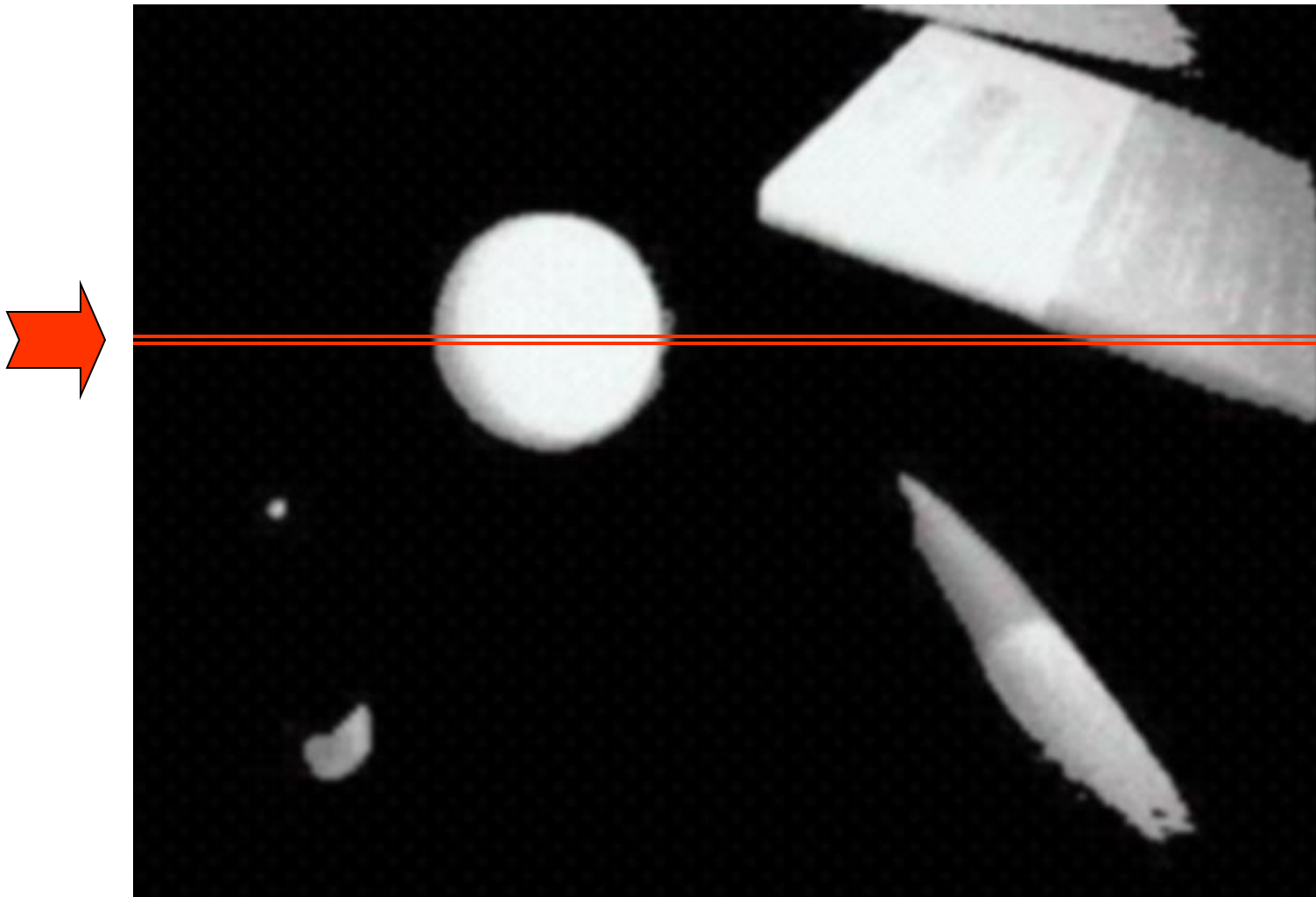


Blur odstraňuje šum, ale nezachováva hrany,  
Bilaterálny filter je pomalý, ale hrany zachováva.  
Iná metóda: anizotropná difúzia (Perona-Malik)

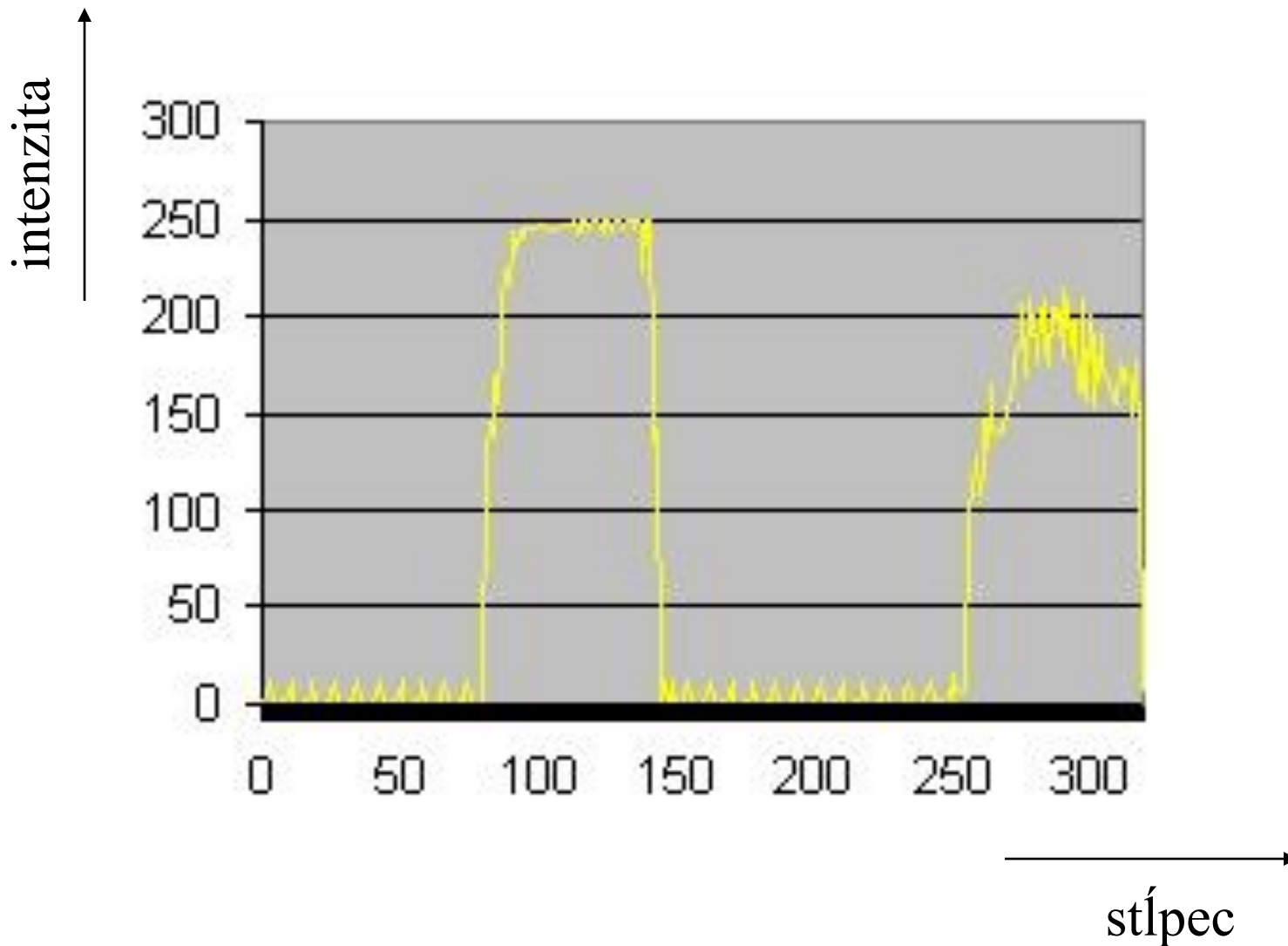


# Hranové operátory

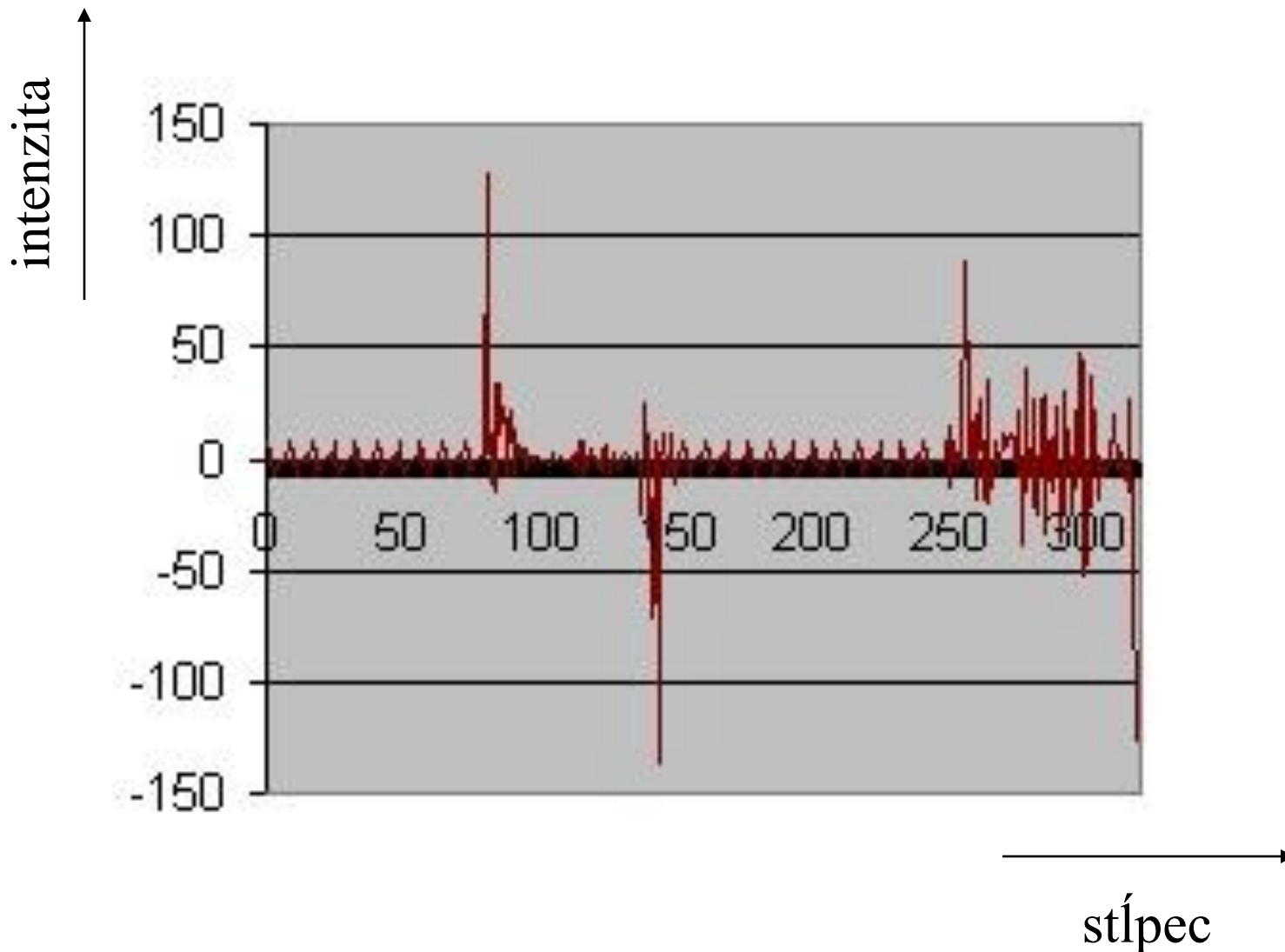
- Sobelov operator
- Laplacian, (LoG, Marr-Hildrethov algoritmus)
- Operátor Canny



- Ked' sa zameriame na jeden riadok



- Hrany zodpovedajú veľkým zmenám  
Ako ich odfiltrujeme?



- Už púhy rozdiel susedných intenzít hrany celkom dobre indikuje

# Sobel kernel (vertikálny)

- Je len o málo sofistikovanejšia metóda
- Uvažuje aj susedné riadky (s menšou váhou) a zvyšuje kompaktnosť hrán rozdielom až od ďalšieho suseda

$a_{i-1,j-1}$	$a_{i-1,j}$	$a_{i-1,j+1}$
$a_{i,j-1}$	$a_{i,j}$	$a_{i,j+1}$
$a_{i+1,j-1}$	$a_{i+1,j}$	$a_{i+1,j+1}$

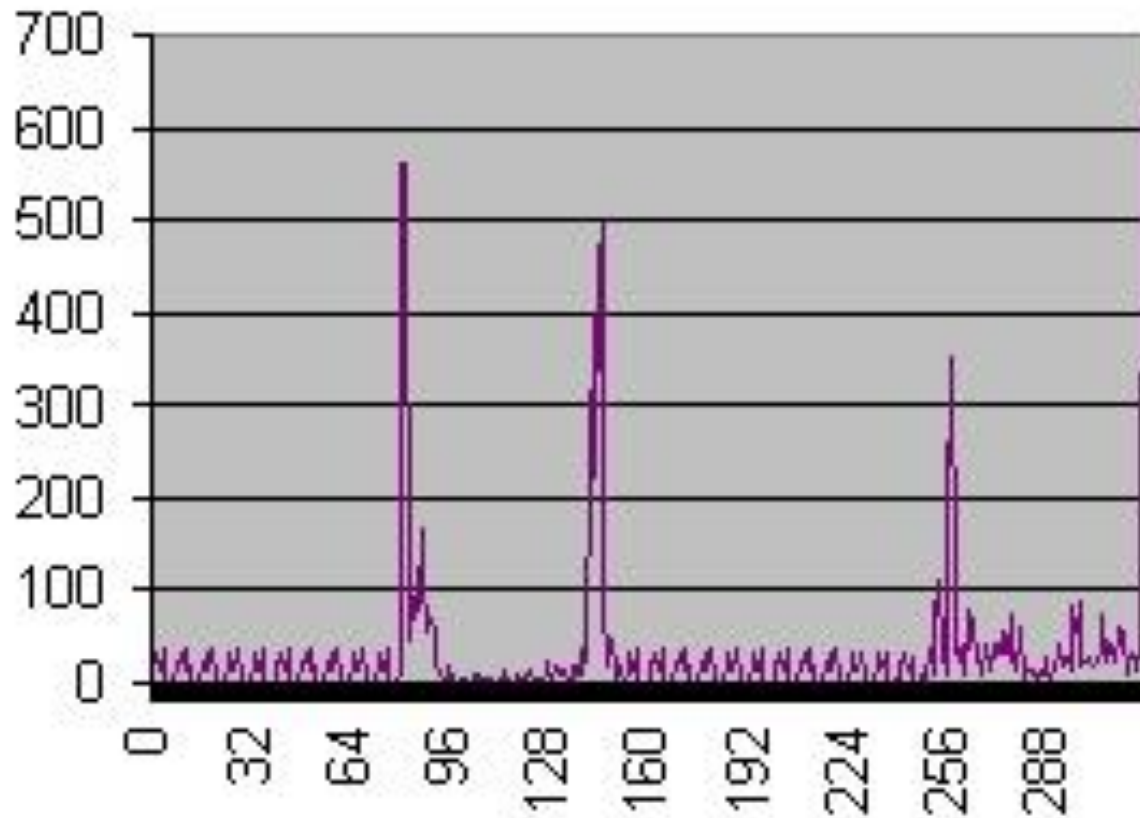
 $\cdot$ 

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

 $=$ 

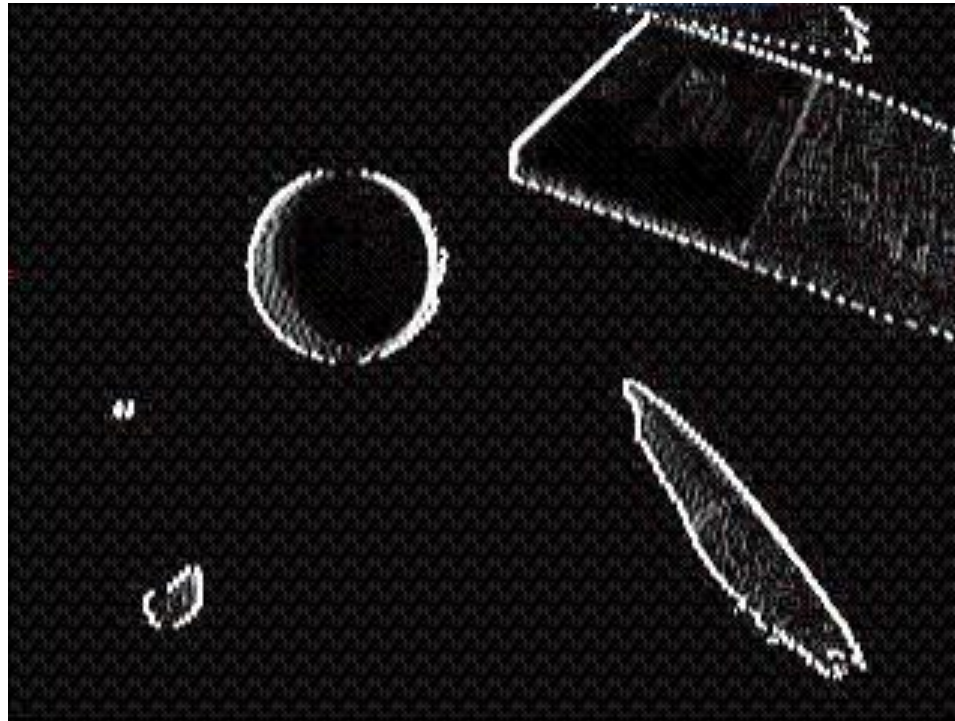
	$b_{i,j}$	

$$b_{i,j} = | a_{i-1,j+1} + 2a_{i,j+1} + a_{i+1,j+1} - a_{i-1,j-1} - 2a_{i,j-1} - a_{i+1,j-1} |$$



Sobelov operátor aproximuje parciálnu deriváciu obrazu  
Dáva lepšie výsledky než púhy rozdiel  
Pozrime sa na výsledný obraz:

# Sobel operator (vertikálny)



$|dx|$

Máme pekné vertikálne hrany, ale takmer žiadne horizontálne. Čo s tým?

# Sobel kernel (horizontálny)

- urobíme to isté zrotované o  $90^\circ$
- A skombinujeme výsledky

$a_{i-1,j-1}$	$a_{i-1,j}$	$a_{i-1,j+1}$
$a_{i,j-1}$	$a_{i,j}$	$a_{i,j+1}$
$a_{i+1,j-1}$	$a_{i+1,j}$	$a_{i+1,j+1}$

 $\cdot$ 

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

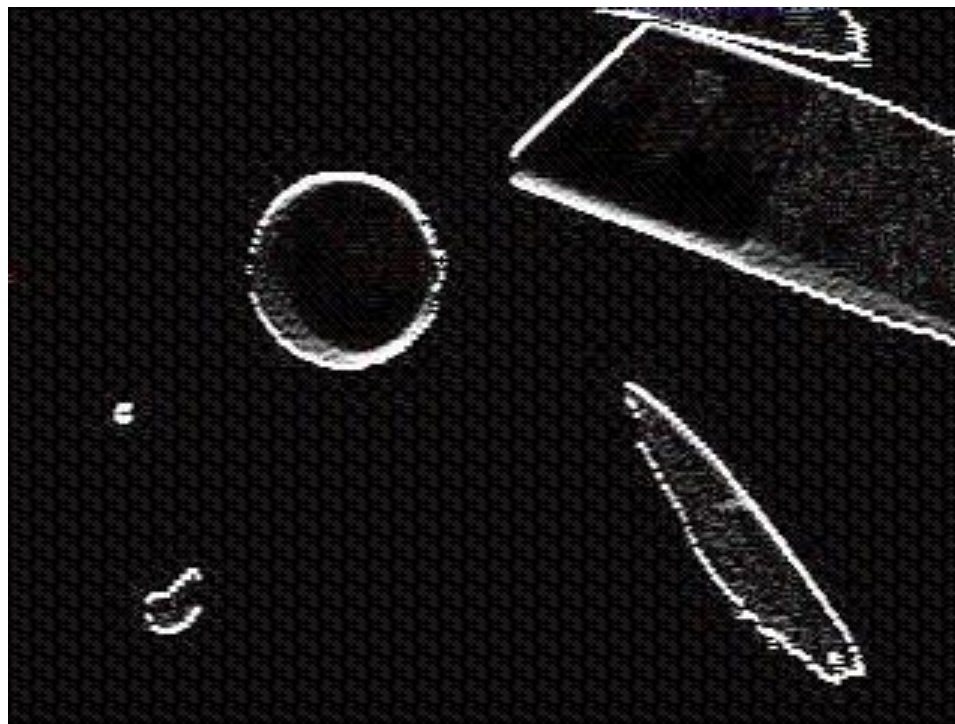
 $=$ 

	$b_{i,j}$	

$$b_{i,j} = | a_{i+1,j-1} + 2a_{i+1,j} + a_{i+1,j+1} - a_{i-1,j-1} - 2a_{i-1,j} - a_{i-1,j+1} |$$



# Sobel operator (horizontálny)

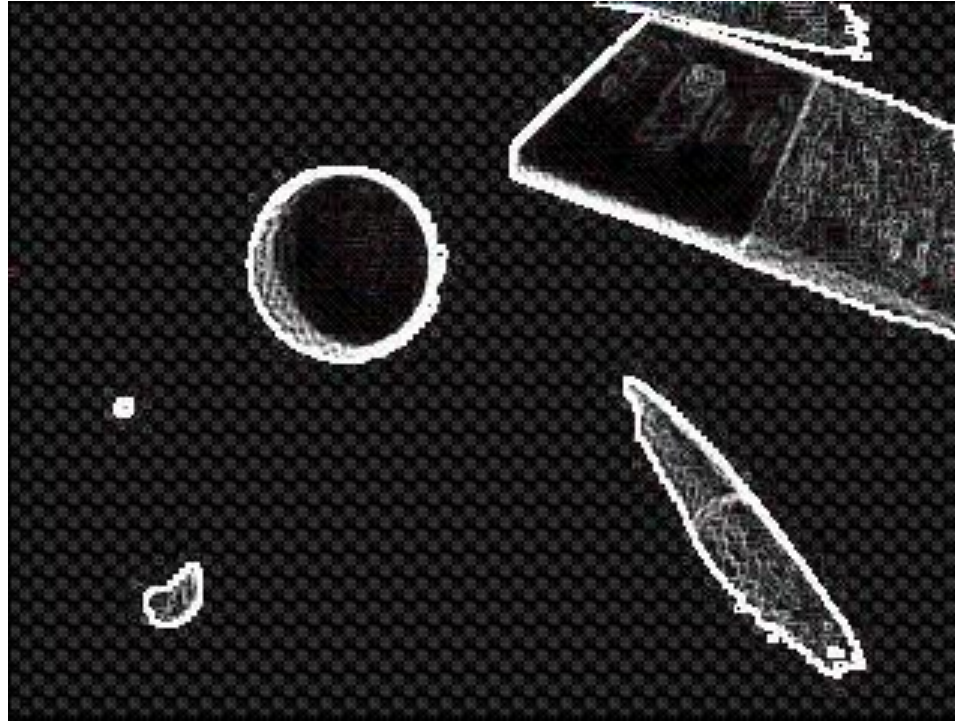


$|dy|$

Aproximuje parciálnu deriváciu v smere y (riadkov)

Dostávame pekné horizontálne hrany

# Sobel operator (magnitúda)



$$|dx| + |dy|$$

or

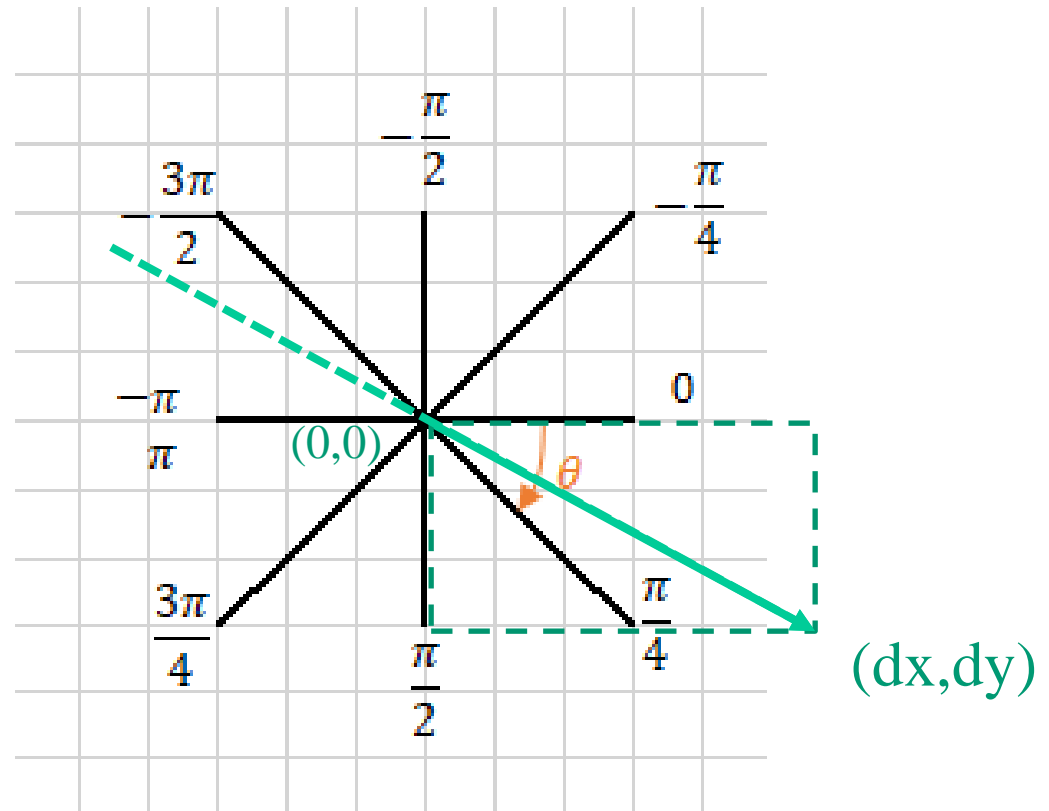
$$(dx^2 + dy^2)^{1/2}$$

Tento kombinovaný výsledok (0-255) musíme ešte prahovať, aby sme dostali hrany (255=hrana, 0=inak)

# Sobel operator (orientácia)

- dx a dy ktoré dáva Sobel operator určujú orientáciu hrán
- $(dx, dy)$  ... gradient

$$\theta = [ \arctan(dy/dx) ]$$



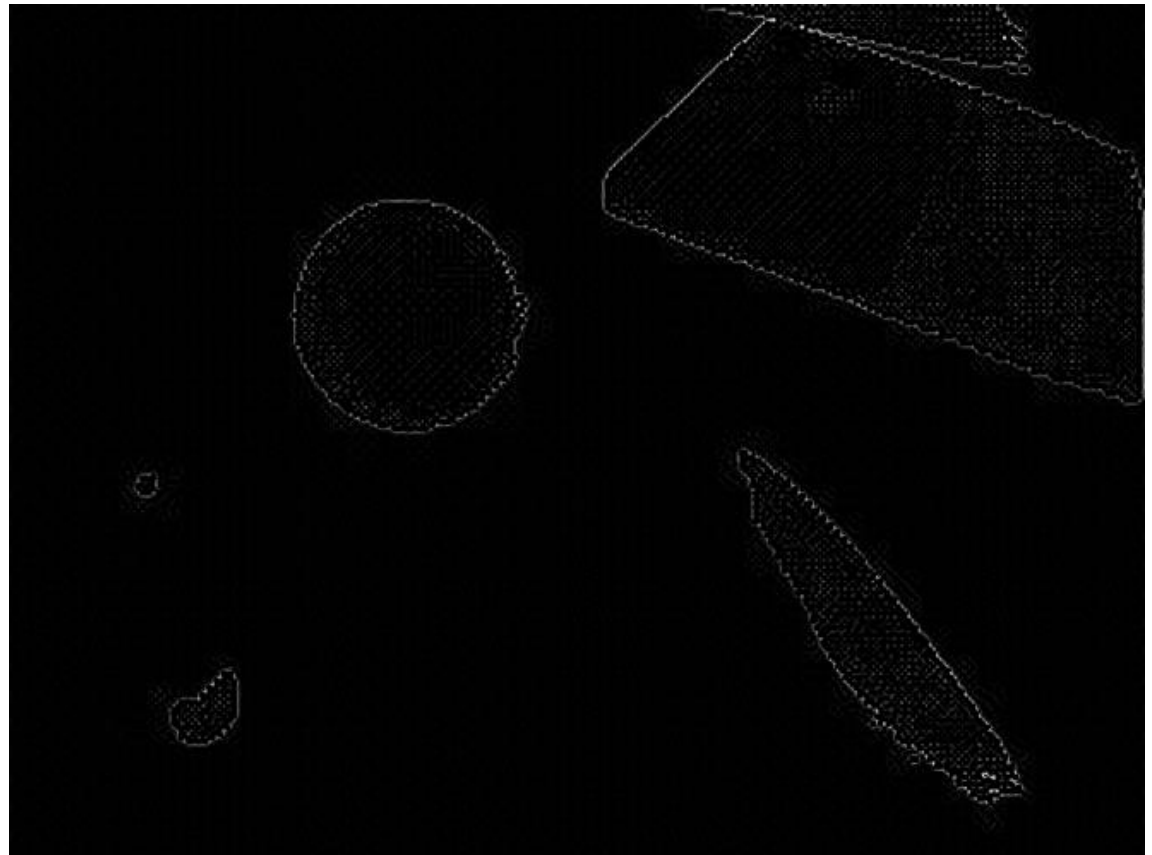
# Laplacian (diskrétny)

$$\Delta = \nabla \nabla$$

$$\nabla = (d/dx, d/dy)$$

$$\Delta_D \dots$$

0	-1	0
-1	4	-1
0	-1	0



```
lap = cv2.Laplacian(frame, cv2.CV_8U)
```

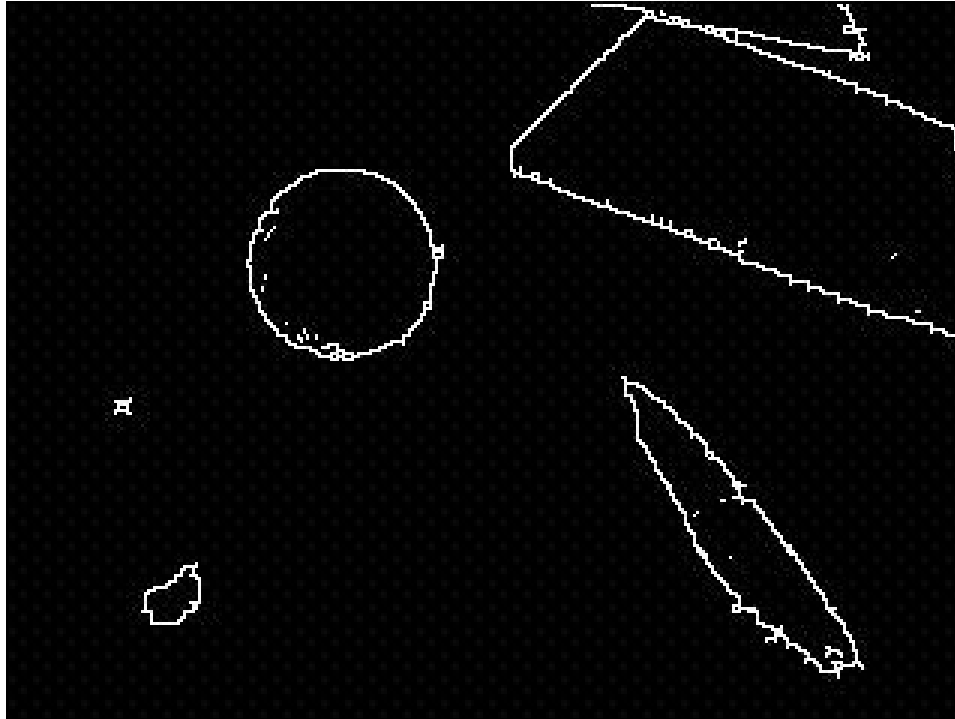
# Výhody a nevýhody

- Sobel dáva hrany trochu (viacmenej) spojité a hrubé, dajú sa stenčovať, orezávať a pod.
- Laplacian dáva hrany tenké ale nespojité
- Obe nevýhody odstraňuje Canny

# Operátor Canny

- Vychádza zo Sobelovho operátora a binarizuje jeho výstup
- Stenčuje hrany (thinning) cez *non-maximum suppression* (zoberie pixely s vyššou intenzitou než sú susedné v smere orientácie hrany, magnitúda ostatných je položená na nulu)
- Aplikuje dva prahy – nižší a vyšší:  
pixel s magnitúdou nad vyšším prahom je vždy hrana, zatiaľ čo pixel medzi nižším a vyšším prahom je hrana, ak susedí s pixelom už vzatým do hrán (hysterézia)

# Operátor Canny



- Takto dostávame v celku kvalitné hrany