

Detekcija ivica

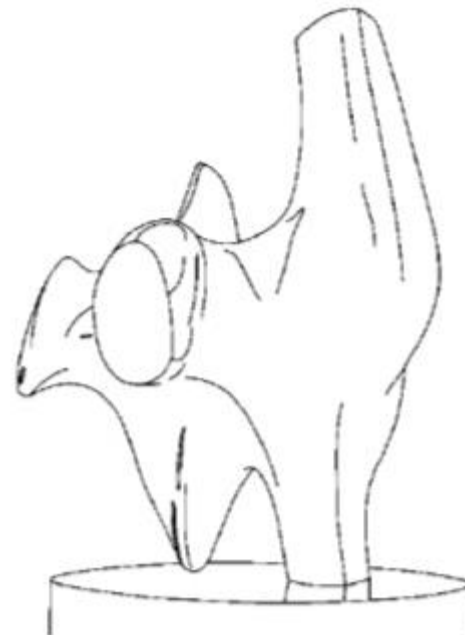
# Detekcija reprezentativnih obeležja

---

- Cilj: identifikovati značajna obeležja na slici, odnosno važne lokalne oblike koji će nam pomoći da prepoznamo oblike na slici
- Primeri ovakvih obeležja:
  - Ivice
  - Geometrijski oblici (linije, kružnice, lukovi,...)
  - Uglovi
  - Domenski specifični oblici
  - Povezani regioni

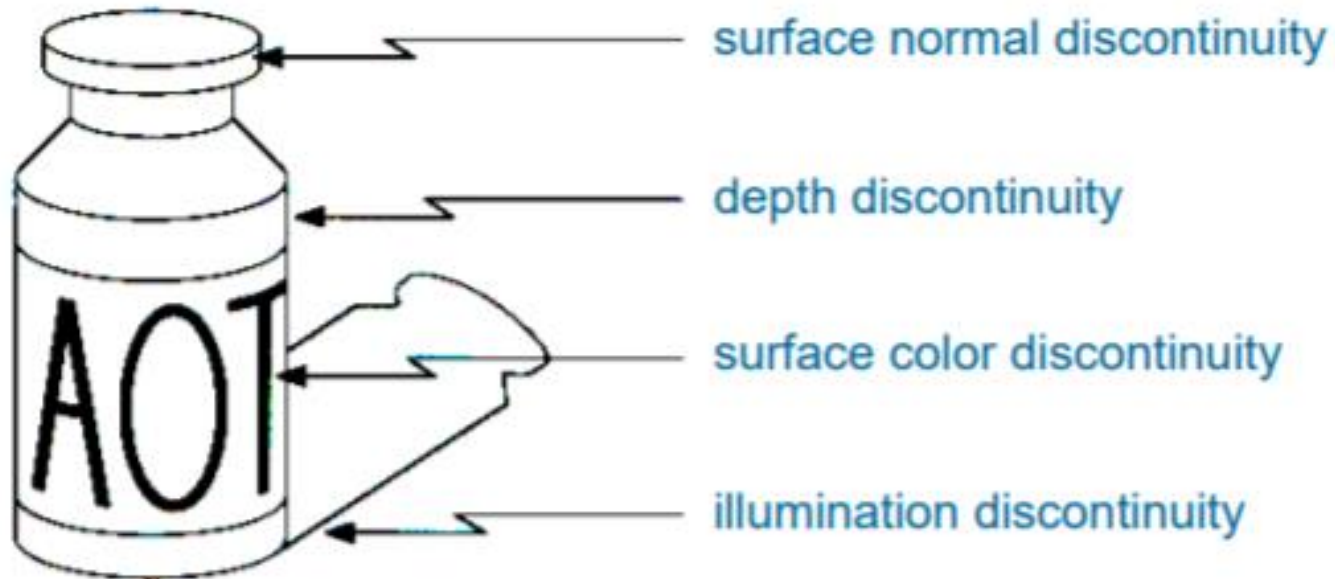
# Detekcija ivica

- Cilj: konvertovati 2D sliku u skup krivih
  - Ovo nam omogućava da pronađemo istaknuta obeležja scene
  - Kompaktnija reprezentacija od piksela
- Pogled iz teorije informacija (*information theory*): ivice kodiraju promenu, a promene su teške za predviđanje. Zato, ivice efikasno kodiraju sliku



# Detekcija ivica

- Ivice se na slici pojavljuju iz različitih razloga



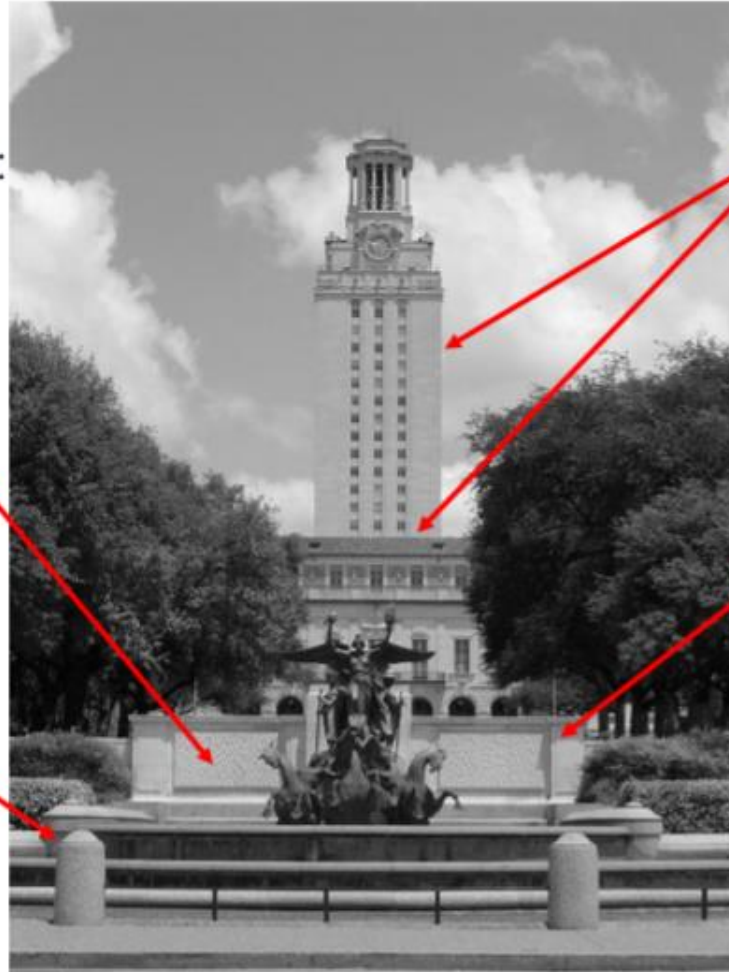
# Detekcija ivica

Reflectance change:  
appearance  
information, texture

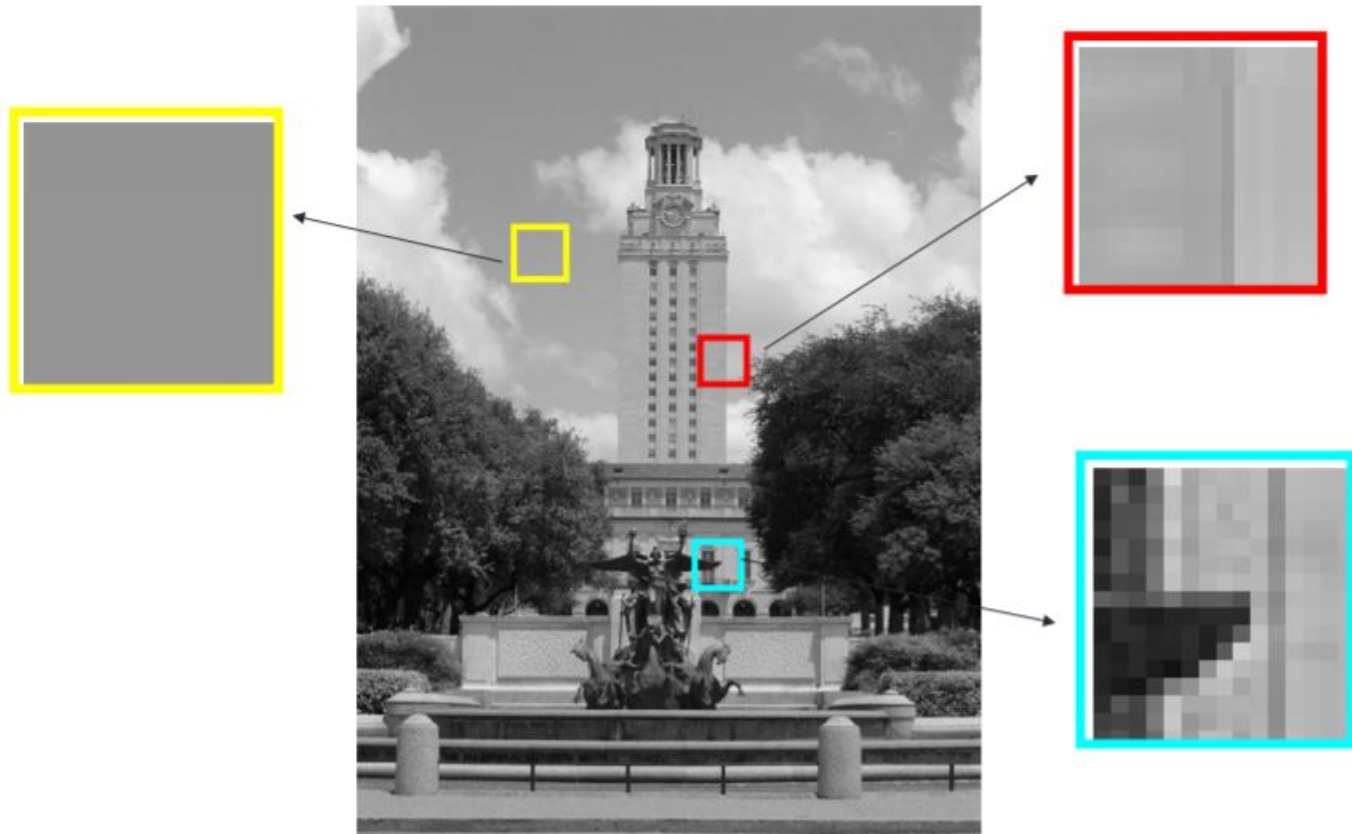
Depth discontinuity:  
object boundary

Change in surface  
orientation: shape

Cast shadows



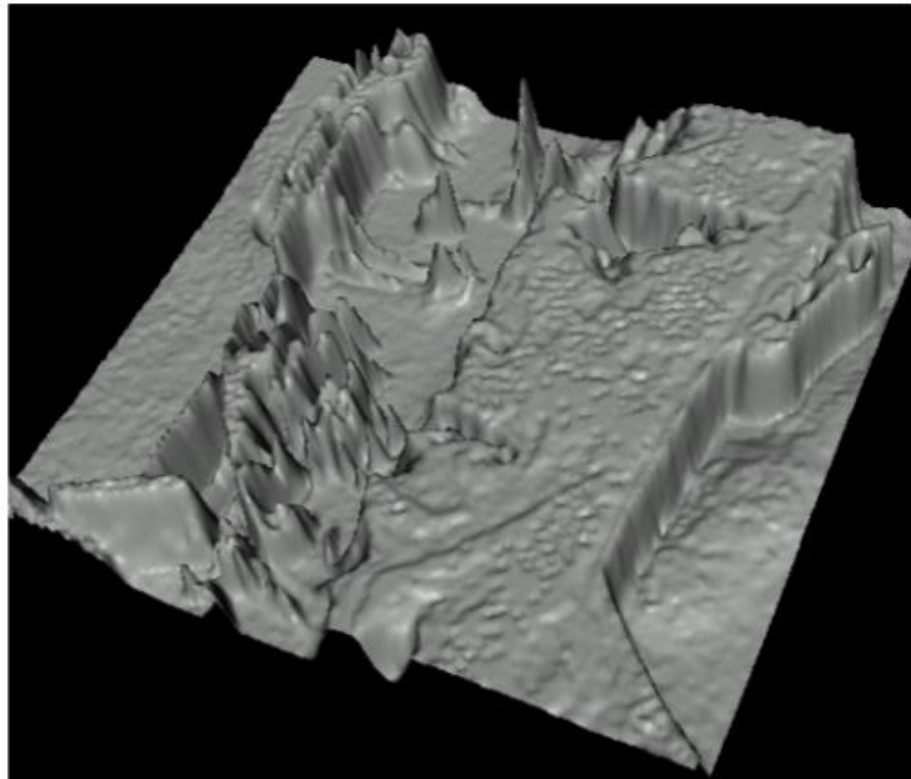
# Kontrast



- Bolje je koristiti lokalna obeležja za detekciju ivica

# Detekcija ivica bazirana na intenzitetu piksela

- Slike smo predstavili kao funkciju intenziteta piksela
  - Ivice su kao oštre litice
  - To su mesta na slici gde ima velikih promena intenziteta na susednim pikselima



# Detekcija ivica bazirana na intenzitetu piksela

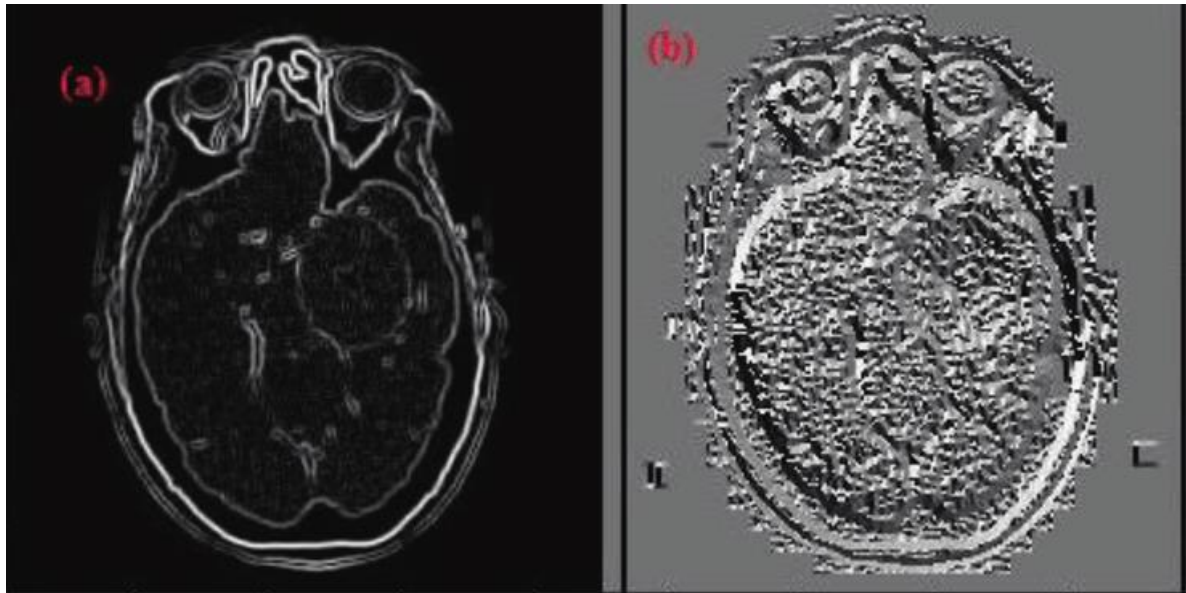
---

- Pri detekciji se oslanjamo samo na intenzitet – ovo nije savršeno
- Ako bi se dva objekta istog intenziteta preklapala, ovo bi nas sprečilo da detektujemo ivicu
  - Morali bismo da koristimo sofisticiranija obeležja poput teksture ili boje
- Međutim, veoma retko se dešava da su dva objekta na slici baš istog intenziteta, pa je intenzitet dobar indikator ivica



# Detekcija ivica

- Ivica je mesto brze promene funkcije intenziteta
- Osnovna ideja: potražiti susedstvo sa značajnim znacima promene



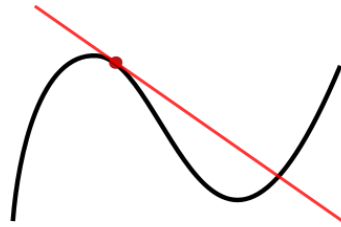
81	82	26	24
82	33	25	25
81	82	26	24

Kako definisati  
susedstvo?

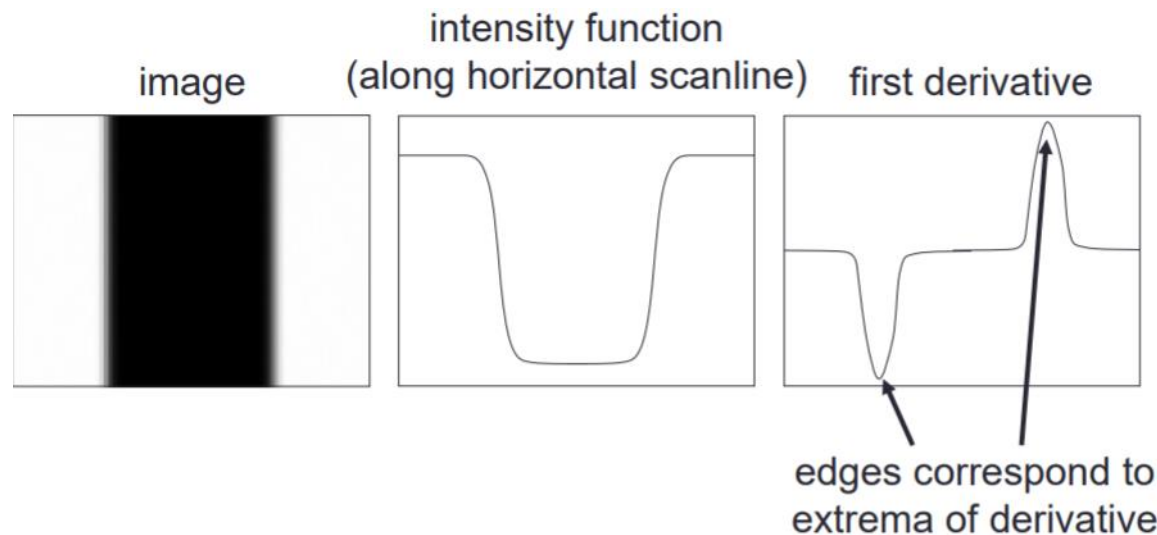
Kako detektovati  
promene?

# Detekcija ivica

- Izvod funkcije oslikava brzinu promene funkcije
  - izvod predstavlja nagib tangente na funkciju – u delovima gde se ona naglo menja, nagib je velik, dok je na delovima gde se sporo menja (blizu lokalnih ekstrema) nagib blizu 0

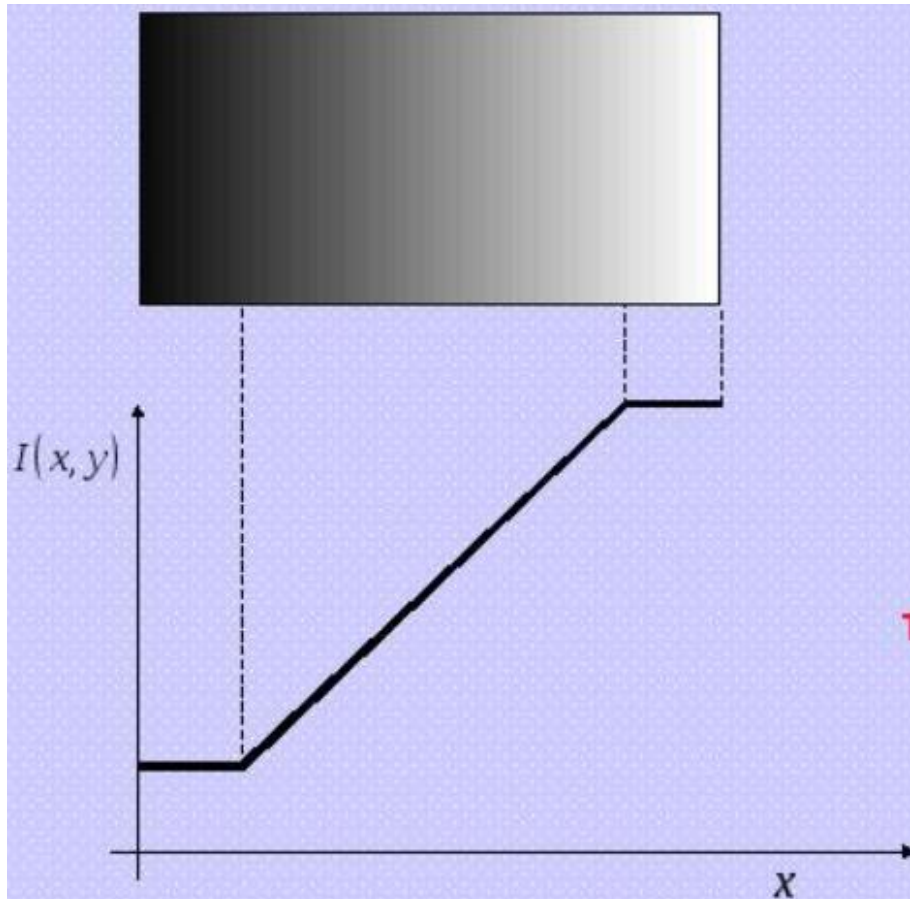


- Dakle, dobra indikacija ivice (nagle promene intenziteta piksela) jeste izvod funkcije intenziteta

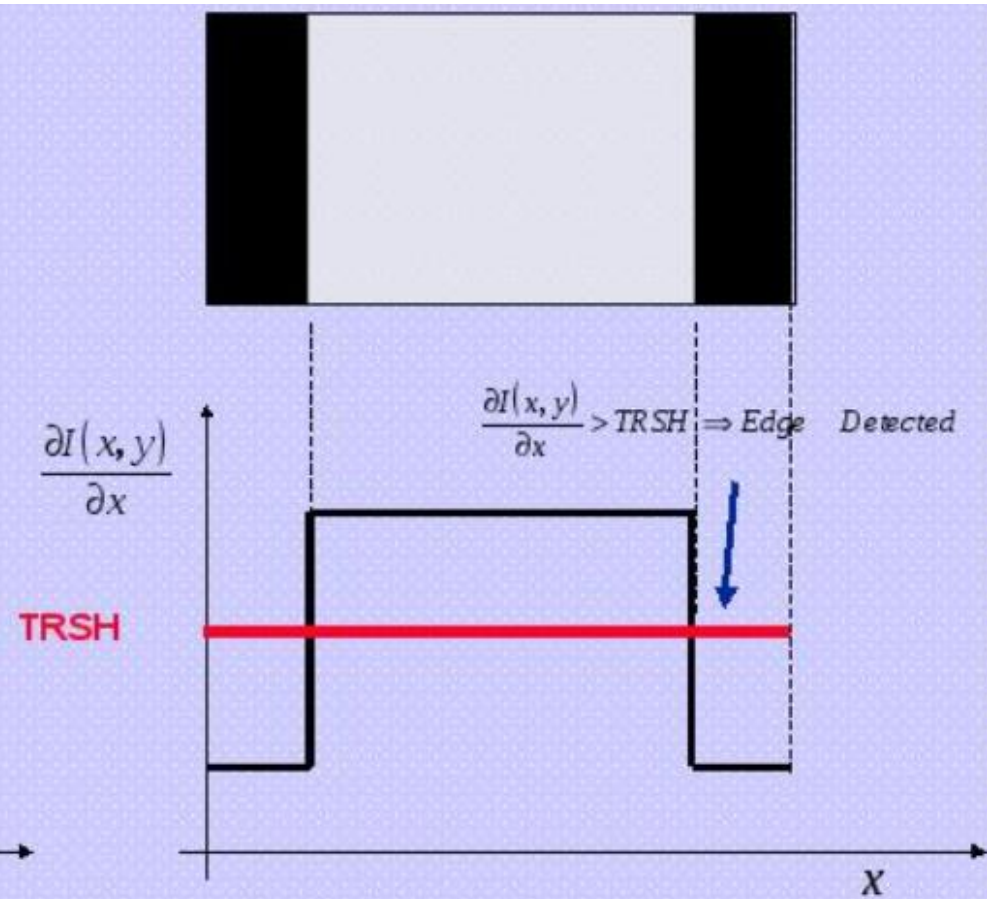


# Detekcija ivica

Funkcija intenziteta



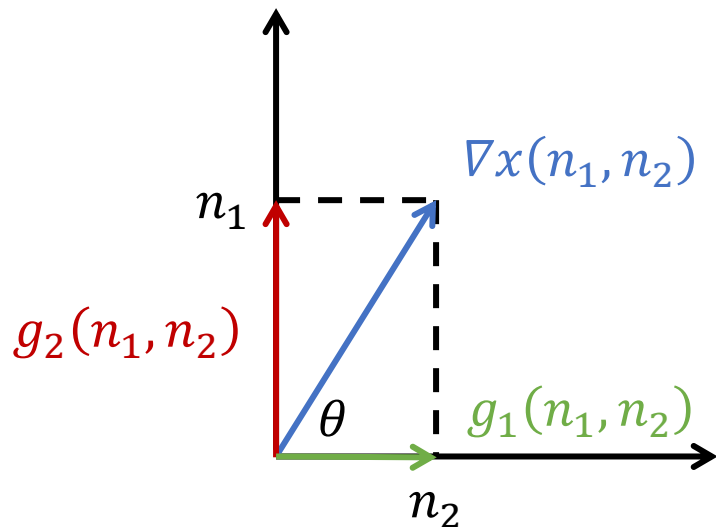
Prvi izvod



# Gradijent slike

- Operator gradijenta

$$g_x(n_1, n_2) = \begin{bmatrix} \frac{\partial x(n_1, n_2)}{\partial n_1} \\ \frac{\partial x(n_1, n_2)}{\partial n_2} \end{bmatrix} = \begin{bmatrix} g_1(n_1, n_2) \\ g_2(n_1, n_2) \end{bmatrix}$$



Magnituda gradijenta:

$$\begin{aligned} M(n_1, n_2) &= |g_x(n_1, n_2)| \\ &= \sqrt{(g_1(n_1, n_2))^2 + (g_2(n_1, n_2))^2} \end{aligned}$$

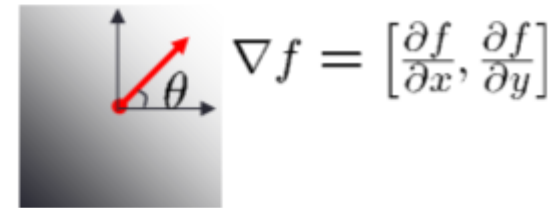
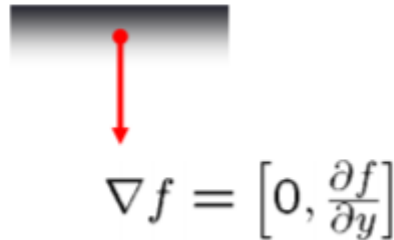
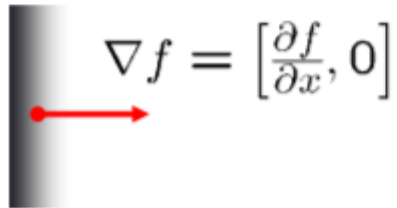
Pravac gradijenta:

$$\theta = \tan^{-1} \left[ \frac{g_2(n_1, n_2)}{g_1(n_1, n_2)} \right]$$

# Detekcija ivica

- Gradijent slike ukazuje na smer najveće promene u intenzitetu piksela

$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$



- Smer gradijenta se računa kao  $\theta = \text{atan} \left( \frac{\partial I}{\partial x} / \frac{\partial I}{\partial y} \right)$  - smer gradijenta je normalan na ivicu

# Detekcija ivica

Magnituda gradijenta je “snaga” ivice

$$\|\nabla f\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Često koristimo kvadratni gradijent da izbegnemo računanje korena

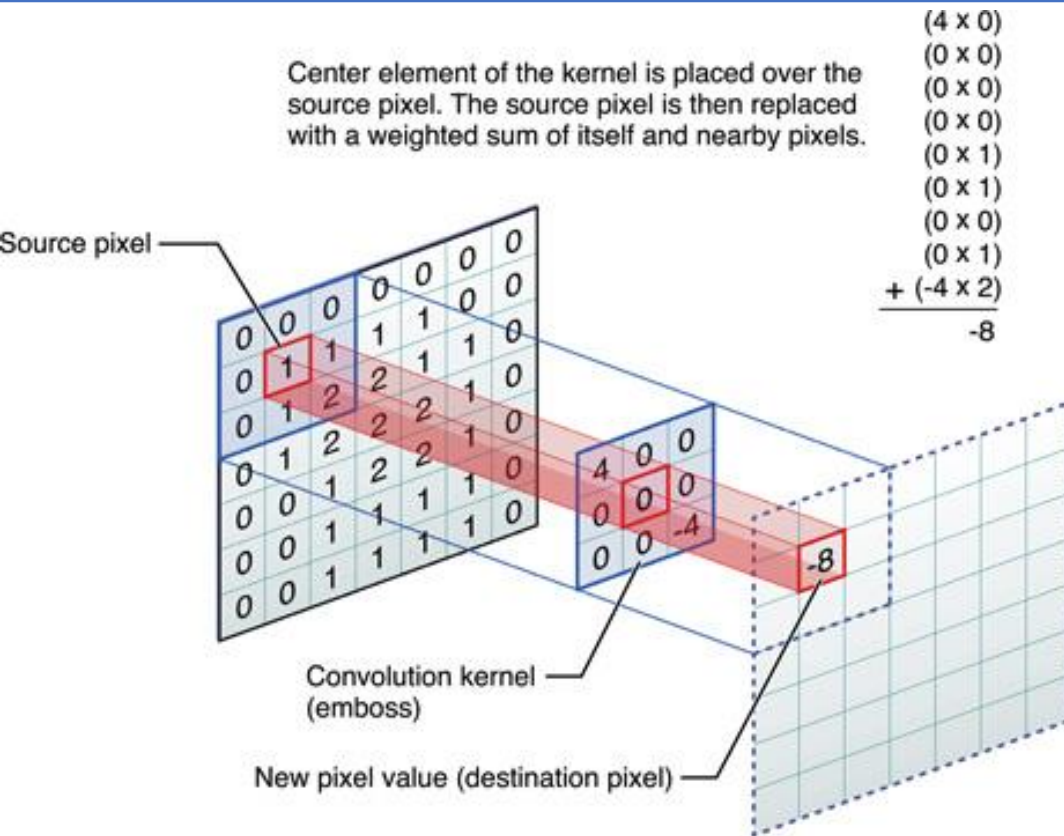


**Blurred**



**Gradient Magnitude**

# Postupak



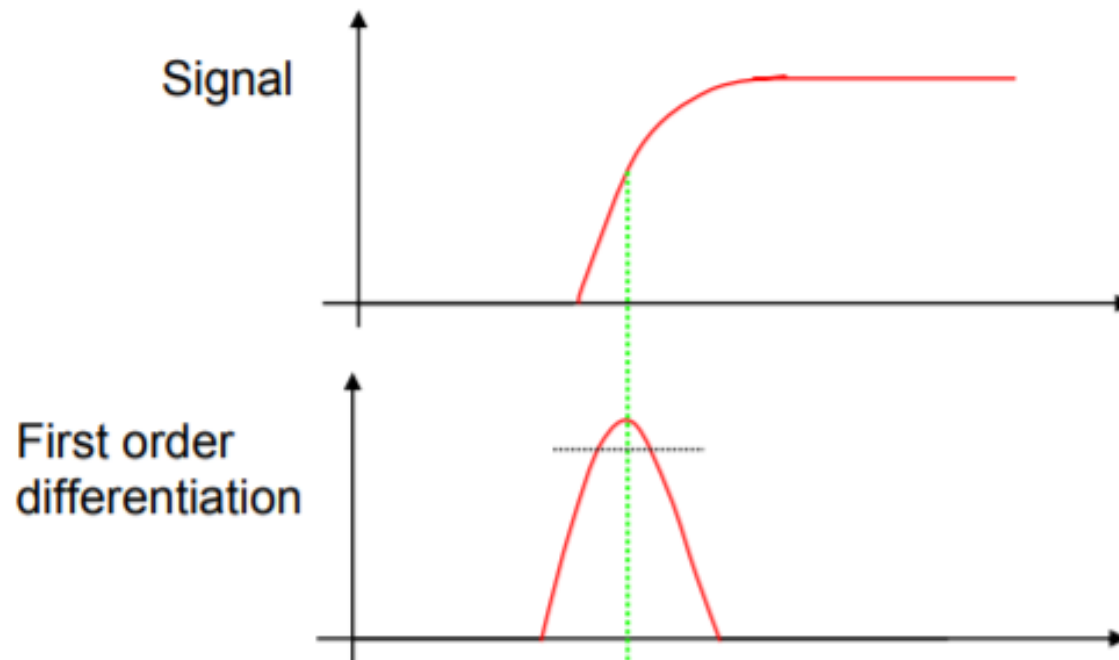
## 1. Primenićemo diferencijalni operator

(kernel koji primenjen na sliku vraća funkciju izvoda te slike)

Idealno, rezultat je 0 tamo gde nema ivice, a velika vrednost na mestima velike promene gradijenta

# Postupak

2. Primenićemo prag na dobijenu funkciju izvoda da selektujemo piksele ivice





# Detekcija ivica – konačne razlike

- Problem: gradijent je definisan za kontinualne funkcije, a funkcija slike je definisana za niz diskretnih vrednosti (određen rezolucijom)
- Definicija parcijalnog izvoda za 2D funkciju:

$$\frac{\partial I(x, y)}{\partial x} = \lim_{\varepsilon \rightarrow 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}$$

- U slučaju diskretnih podataka, aproksimiraćemo ovu formulu metodom konačnih razlika:

$$\frac{\partial I(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x, y)}{1} = f(x + 1, y) - f(x, y)$$

1	0	1	0	10	11	11	0	1
---	---	---	---	----	----	----	---	---

-1	1	-1	10	1	0	-11	1	0
----	---	----	----	---	---	-----	---	---

# Detekcija ivica – Roberts Cross

- Određivanje ivica metodom konačnih razlika možemo sprovesti pomoću konvolutivnih filtera:

1	-1	1
-1		

$$G_x = f(x + 1, y) - f(x, y)$$

$$G_y = f(x, y) - f(x, y + 1)$$

- Kerneli su jednaki, samo zarotirani za  $90^\circ$
- Iako ovo ne odgovara tačno izvodu po  $x$  i  $y$ , odgovara izvodu po diagonalnim pravcima
- Kerneli će imati maksimalan *response* na ivice pod uglom od  $45^\circ$

+1	
	-1

$g_1$

	+1
-1	

$g_2$

Roberts Cross convolution kernels

# Roberts Cross

+1	
	-1

$g_1$

	+1
-1	

$g_2$

Roberts Cross convolution kernels

- Kerneli se primenjuju odvojeno na ulaznu sliku
  - odvojeno se računaju komponente gradijenta za svaki pravac
- Zatim se rezultat kombinuju da se pronade magnituda izvoda:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

(tipično aproksimiramo sa  $|G| = |G_x| + |G_y|$  jer je brže za računanje)

- Ugao/orijentacija ivice se računa kao  $\theta = \text{atan} \frac{G_x}{G_y} - \frac{3\pi}{4}$

$$|G| = \sqrt{G_x^2 + G_y^2} \quad |G| = |G_x| + |G_y|$$



original image



+1	
	-1

$g_1$



	+1
-1	

$g_2$

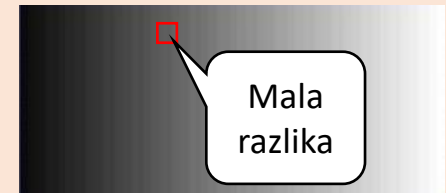
# Roberts Cross – prednosti i mane

## Mali kernel

- Brz: za izlaz koristimo samo 4 piksela ulaza i koristimo samo + i –
- Nema parametara koje treba podešavati

- Veoma osetljiv na šum: posmatra razliku na 1 pikselu

- Slab odziv na ivice koje nisu oštre



- Nejasno koji izlazni piksel odgovara kom ulaznom pikselu (slika gradijenta pomerena za pola piksela i po  $x$  i  $y$  u odnosu na ulaznu)

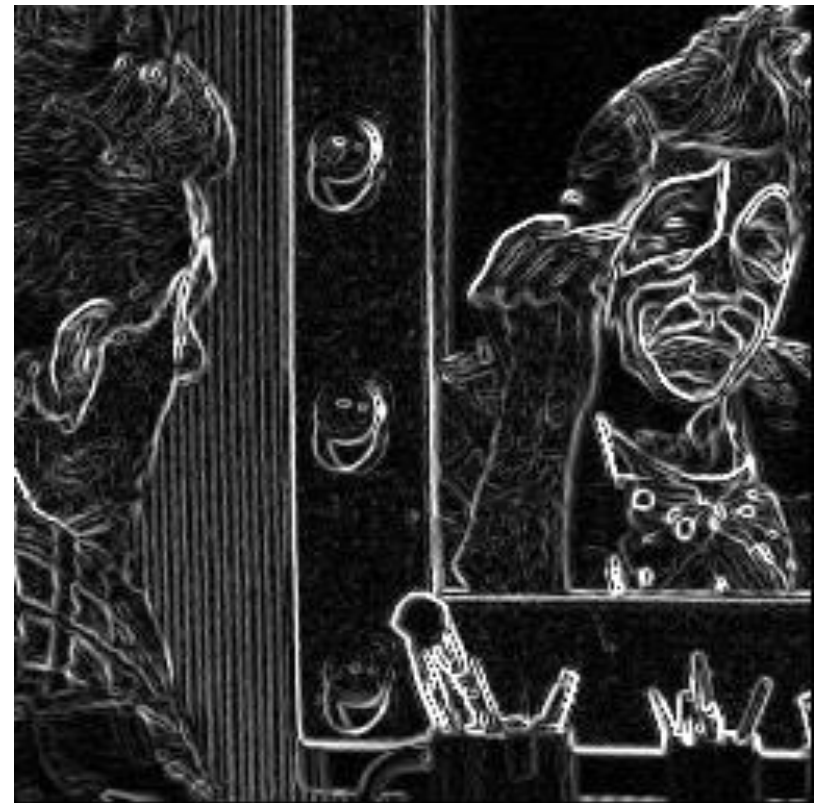
# Roberts Cross – osetljivost na šum

(Da bi slika bila jasnija, izlaz operatora je pomnožen sa 5)



# Rešenje

Ideja: samo su najsnažnije ivice pouzdane - primenimo prag da dobijemo binarnu sliku

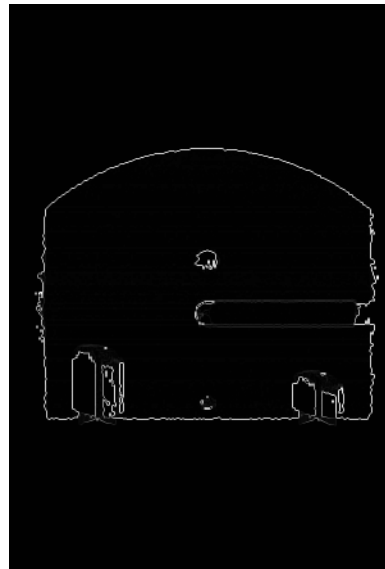


# Roberts Cross – osetljivost na šum

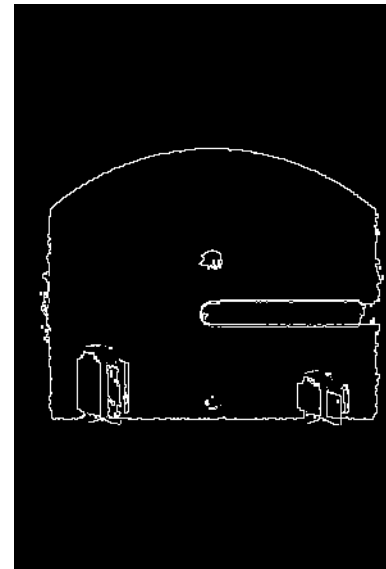
- Možemo ga koristiti za detekciju *depth discontinuity*
  - Distanca objekta od aparata za snimanje je enkodirana intenzitetom piksela
  - Operator rezultuje linijom velikog intenziteta na granicama slike



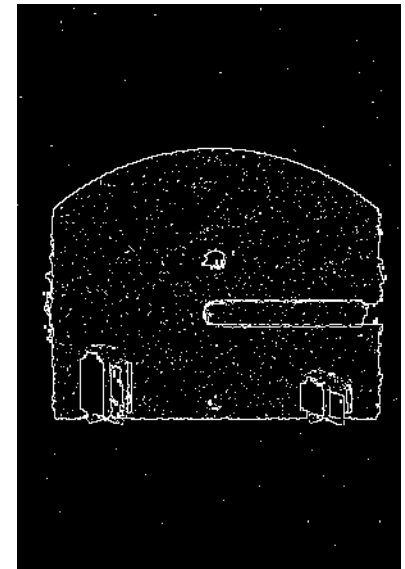
Original



Slabija percepcija  
*depth*  
*discontinuity*  
unutar objekta



Manji prag (20)  
pomaže



Ali ako dodamo  
šum i on će biti  
uključen u izlaz



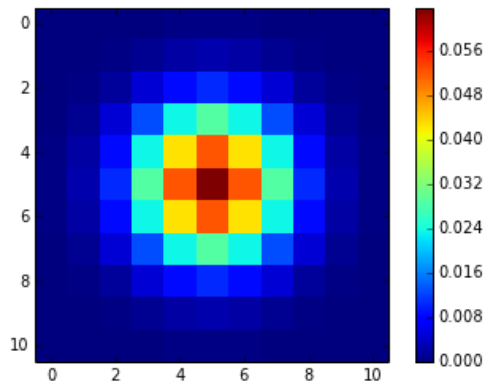
# Gaussian smoothing

- Šum možemo rešiti izглаđivanjem (*smoothing*)

noisy



Gaussian filter

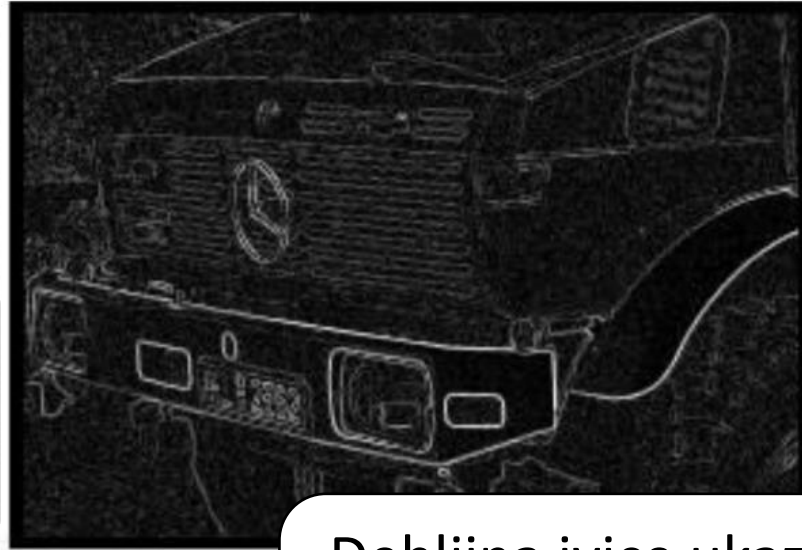


- Često se koriste  $3 \times 3$  ili  $5 \times 5$  Gausovi filteri
- Značajno je smanjen šum, ali slika je smanjene oštrine

# Efekat izgladivanja na Roberts Cross



Pretprocesiranje  
izgladivanjem je dovelo  
do boljeg razlikovanja  
ivica od šuma



Original



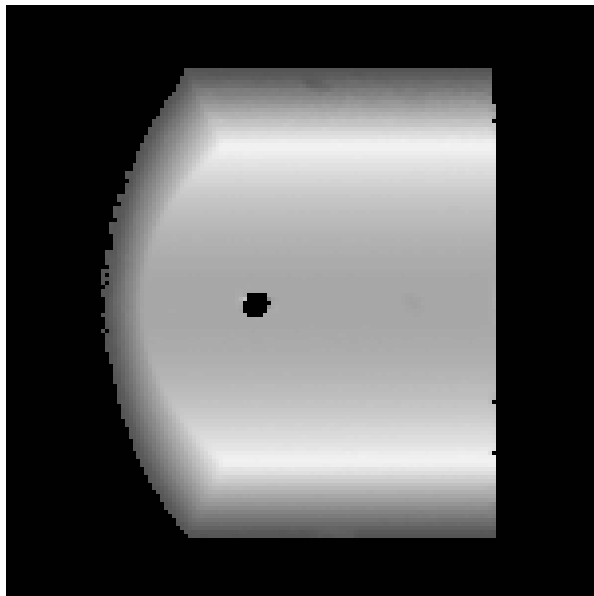
Debljina ivice ukazuje na  
nesigurnost gde se tačna  
ivica nalazi



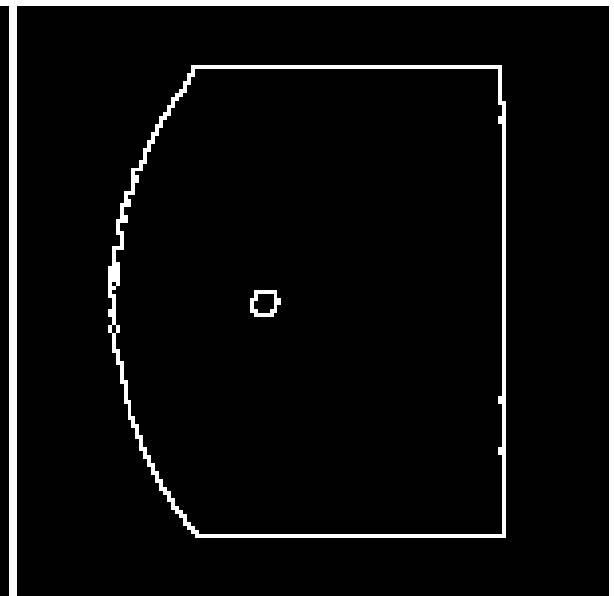
$5 \times 5$   
Gaussian  
filter

# Roberts Cross – oštre ivice

- Najbolje reaguje na oštre ivice
- Ovo je primer gde ivice nisu tako oštre



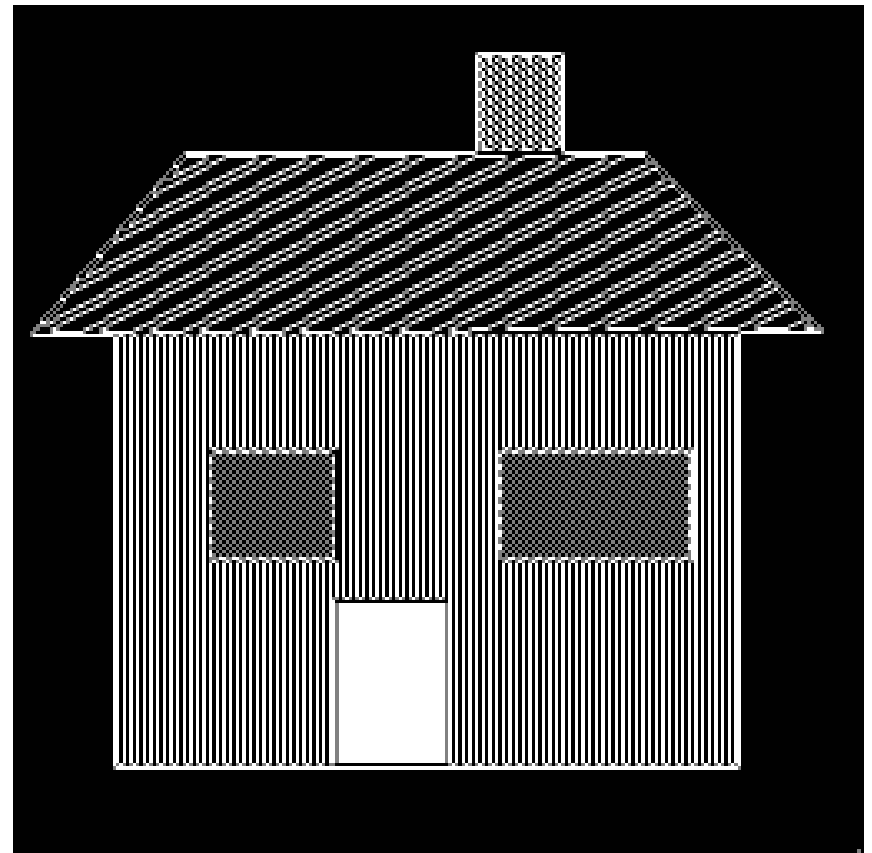
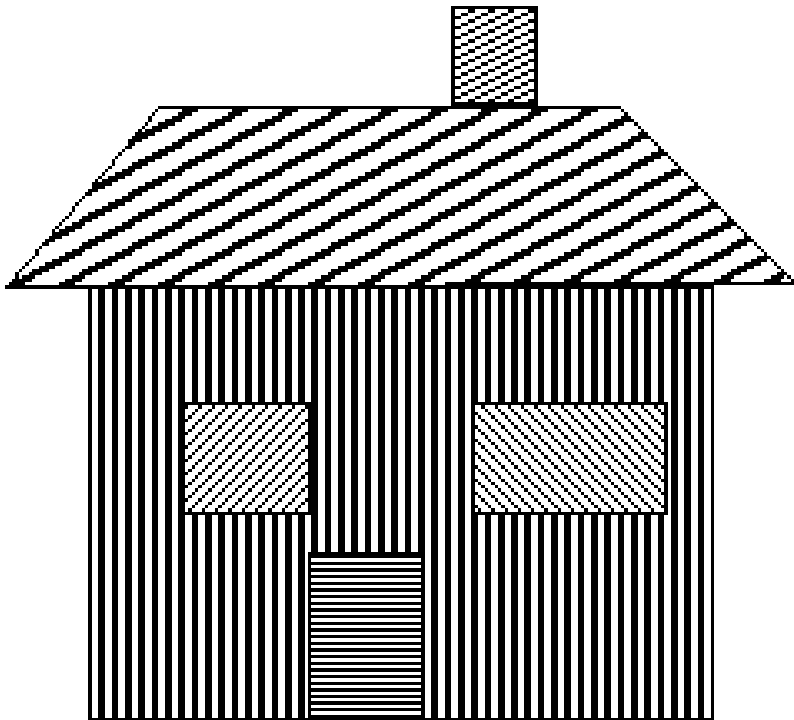
Veoma slabe ivice



Kao posledica, ne možemo ih razdvojiti od šuma

# Roberts Cross

- Uticaj oblika ivica (intenzitet svih ivica na originalu je jednak)
- Različita širina i orijentacija ivica utiču da rezultati veoma variraju



# Sobel

- Sličan je Roberts-Cross operatoru
- Sastoji se od dva kernela koji su jednaki, samo je jedan zarotiran za  $90^\circ$ :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Kerneli su dizajnirani da maksimalno reaguju na ivice koje su vertikalne i horizontalne
- Primenujemo jedan pa drugi kernel da dobijemo dva rezultata koje potom kombinujemo:
  - $|G| = \sqrt{G_x^2 + G_y^2}, \theta = \arctan(G_y/G_x)$

# Sobel – prednosti i mane

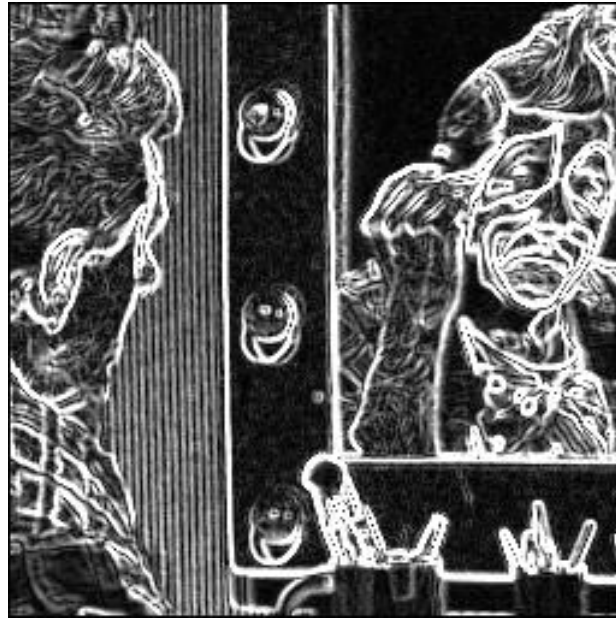
---

- Sporiji je od Roberts Cross
- Veći kernel „izgladjuje“ ulaznu sliku više od Roberts Cross operatora pa je Sobel manje osetljiv na šum
- Prirodne ivice često rezultuju izlaznim ivicama koje su par piksela široke zbog efekta „izgladivanja“
  - Možete primeniti *thinnig* operator (sličan eroziji/otvaranju)

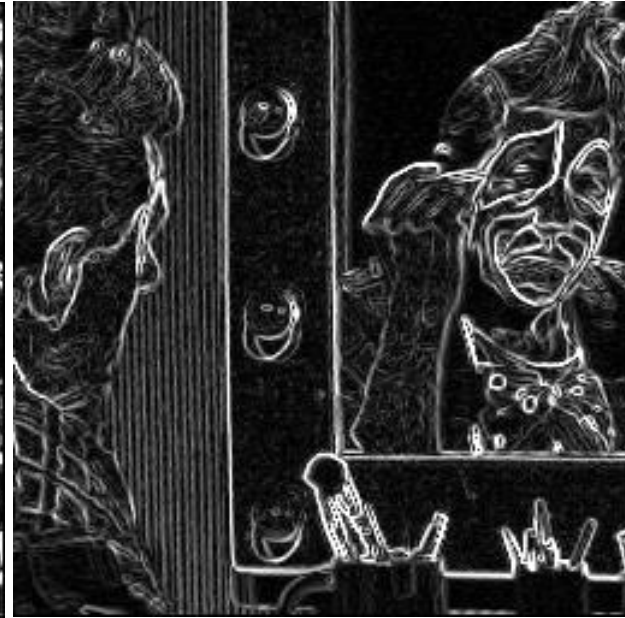
# Sobel i Roberts Cross



Original



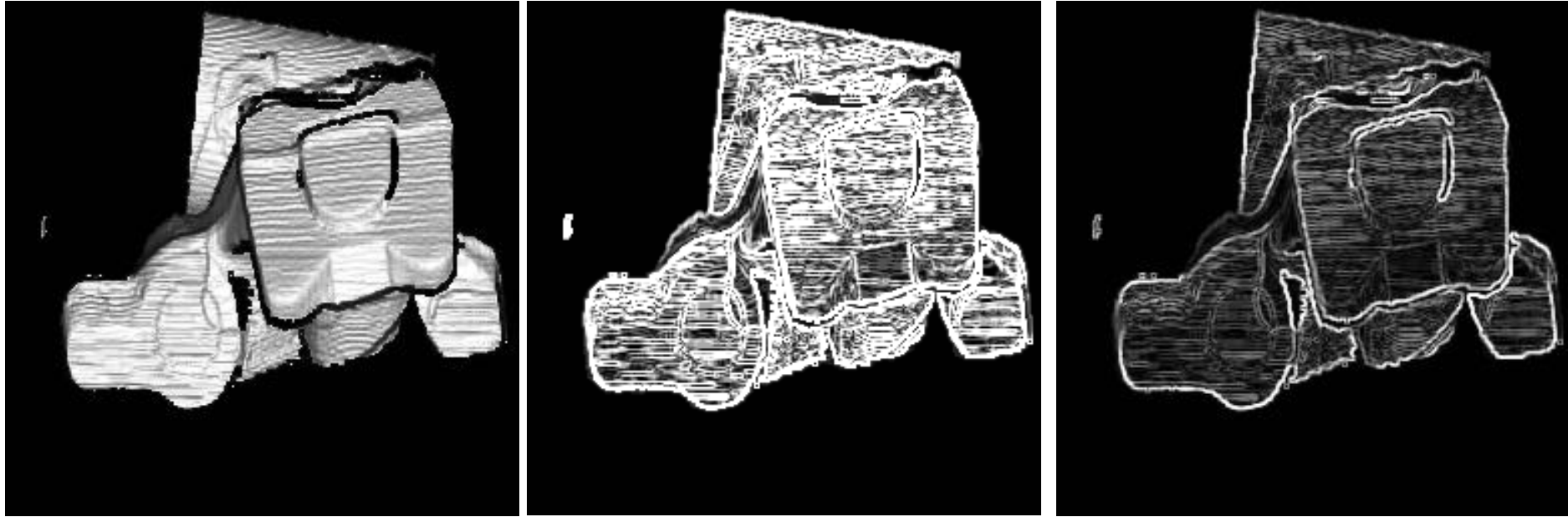
Sobel



Roberts Cross

- Šum koji utiče na Roberts Cross je i ovde prisutan, ali je njegov intenzitet slabiji u odnosu na stvarne ivice
  - Ostavlja mogućnost da se boljim odabirom praga možemo rešiti šuma
- Primetite da su rezultujuće ivice deblje u odnosu na Roberts Cross
  - Posledica izgladivanja (*smoothing*)

# Sobel



- Još nismo sasvim rešili problem kada imamo mnogo finijih varijacija usred dubine
  - Menjanjem praga ne možemo sasvim ukloniti šum
- Canny operator bi radio bolje za ovaj problem



# Prewitt

---

- Veoma sličan Sobelovom operatoru, malo drugačiji kerneli
- Koristi se za detekciju vertikalnih i horizontalnih ivica

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix},$$

# Canny

---

- Dizajniran da bude optimalan detektor ivica
- Dobar detektor ivica:
  - Dobra detekcija – reaguje na ivice a ne na šum
  - Dobra lokalizacija – detektovana ivica je blizu stvarne. Odgovor je minimalan – jedan piksel po ivici
- Imamo nagodbu detekcije/lokalizacije
  - Više izgladivanja pobojšava detekciju, ali lokalizaciju čini gorom

# Canny – postupak

---

## 1. Uklanjanje šuma

- izглаđivanje (*smoothing*) Gausovim operatorom

## 2. Računanje magnitude i orijentacije gradijenta

- Jednostavan operator poput Roberts Cross
- Ivice na ulaznoj slici će rezultovati grebenima na izlaznoj slici

## 3. *Non-maxima suppression*

- Obezbeđuje minimalan odgovor tako što „istanjuje“ linije izlaza
- Ivica se nalazi tamo gde je gradijent najveći
- Algoritam ide po grebenima (*ridge tracker*) i postavlja na 0 sve piksele koji nisu na vrhu grebena

## 4. Primena dva praga

# Canny: 1. korak

---



Original



*Grayscale*



*Gaussian blur*

# Canny: 2. korak

- Računanje intenziteta gradijenata



$G_x$



$G_y$

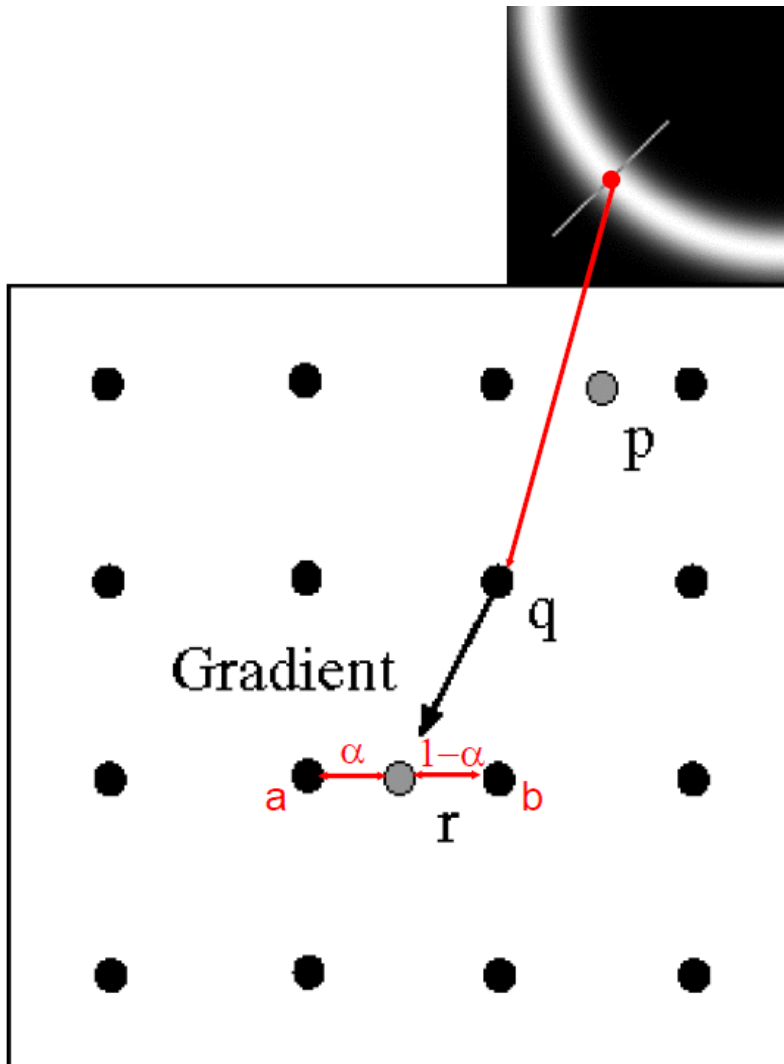
# Canny: 2. korak

- Računanje magnitude i uglova



Magnituda

# Canny: 3. korak – Non-maximum suppression



- Konačna slika bi trebala imati tanke ivice
- Pronalazi se piksel maksimalne vrednosti na ivici
- Piksel  $q$  ima veći intenzitet od piksela  $p$  i  $r$  koji imaju istu orijentaciju (ugao) gradijenta
- $q$  čuvamo, a  $p$  i  $r$  uklanjamo
- Dakle, koristimo sve informacije o gradijentu: magnitudu i ugao

# Canny: 3. korak – Non-maximum suppression



Originalna magnituda



Non-maximum suppression



# Non-maxima supression

---

- Od prethodnog koraka dobićemo tanku ivicu, ali možda ne i konekovanu
  - Ovo je generalan problem sa algoritmima za detekciju ivica baziranim na gradijentu
- Prisutan je i šum
- Problemi se rešavaju primenom *double thresholding*

# Double thresholding

- Pragovi  $T_1$  i  $T_2$  ( $T_1 > T_2$ )
  - Sve iznad  $T_1$  je „jaka“ ivica
  - Sve između  $T_1$  i  $T_2$  je „slaba“ ivica (postoji određena nesigurnost jesu li ili nisu ivice)
  - Pikseli ispod  $T_2$  nisu ivice



# Double thresholding

- Koje slabe ivice su zapravo ivice?
- *Edge tracking*:
  - Slabe ivice koje su povezane sa jakim ivicama su stvarne ivice
  - Slabe ivice koje nisu povezane sa jakim ivicama se uklanjaju



# Canny

---

- Tri parametra: širina Gausovog kernela,  $T_1$  i  $T_2$
- Veća širina Gausovog kernela  $\rightarrow$  manja osetljivost na šum, ali se gube finiji detalji
- Za dobre rezultate, obično biramo visoko  $T_1$  i nisko  $T_2$
- Postavka  $T_2$  na previsoku vrednost može rezultovati da ivice sa šumom budu razbijene u fragmente
- Postavka  $T_1$  na prenisku vrednost može rezultovati zadržavanjem lažnih fragmenata ivica u izlazu

# Canny

---

- Problem su *Y-junctions* (mesta gde se tri grebena sastaju u gradijentu)
  - Dešava se kada je linija delimično zaklonjena drugim segmentom
  - Dve linije će ispasti jedan segment a treća neće biti sasvim povezana sa njima
  - Može se rešti uključivanjem ovog modela u *ridge tracker*

# Canny



Original



Gausov kernel sa  $\sigma = 1$ ,  $T_1 = 255$ ,  $T_2 = 1$

Većina važnih ivica je detektovana, ali još ima previše detalja za dalju obradu

U donjem levom uglu ogledala postoji *Y-junction* problem

# Canny



Previsok  $T_2(220)$  - ivice su rasparčaniye. Vertikalne ivice na zidu nisu detektovane celom dužinom



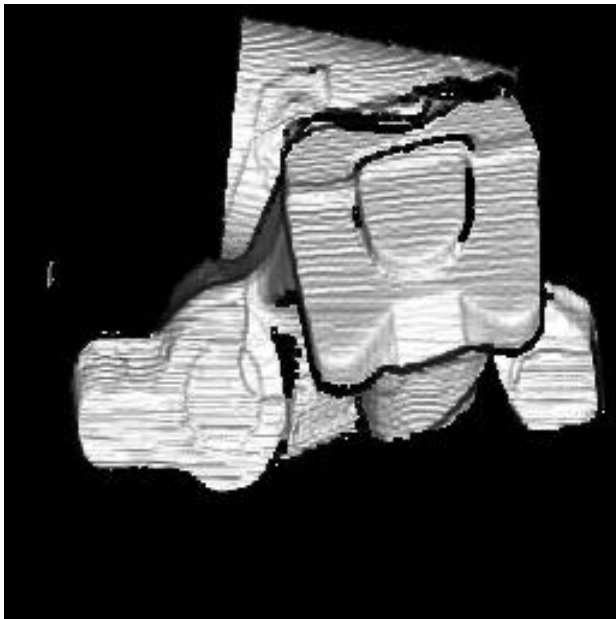
Prenizak  $T_1(128)$  - mnoge slabe ivice detektovane sa fragmentima koji predstavljaju šum

Detalji kose su takođe uhvaćeni

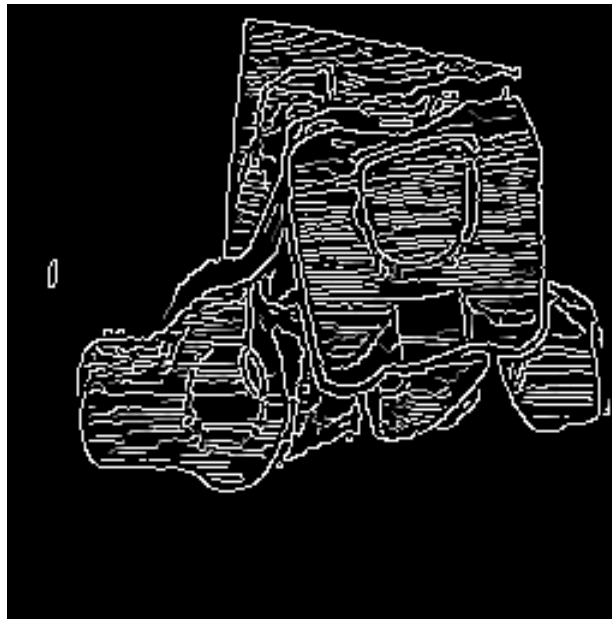


$T_1 = 128, \sigma = 2$   
Ukonjeni detalji zida, ali najsnažnije ivice su tu, glatkije su i ima manje šuma

# Canny



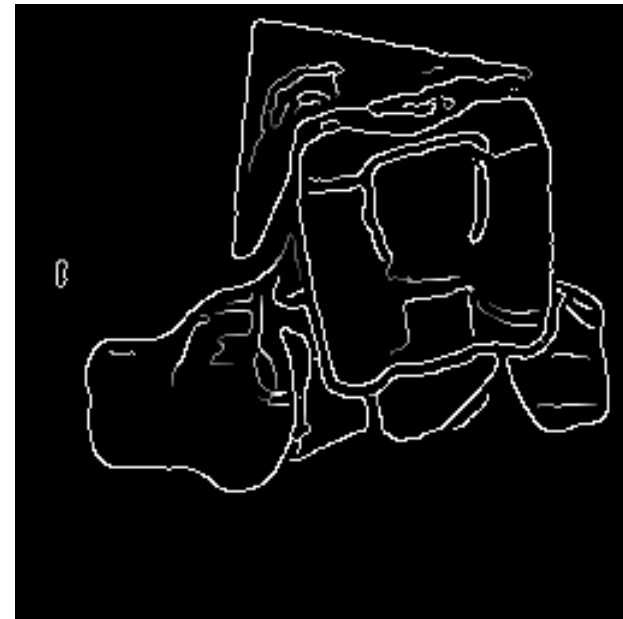
Original



Gausov kernel sa  $\sigma = 1$ ,  $T_1 = 255$ ,  $T_2 = 1$

Mnogo detalja

Ali nije dobro ako smo  
zainteresovani samo za spoljne  
ivice



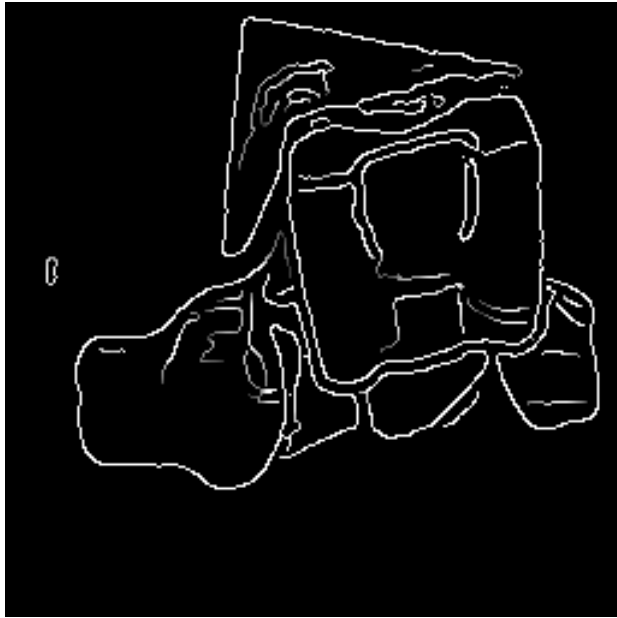
Gausov kernel sa  $\sigma = 1.8$ ,  $T_1 = 255$ ,  $T_2 = 1$

Neke ivice koje su nastale usled  
promene orijentacije površine su  
ostale, ali su slabije od okvira –  
regulišemo pragovima

- Gausov filter može da posluži za dve svrhe: kontrola nivoa detalja i uklanjanje šuma

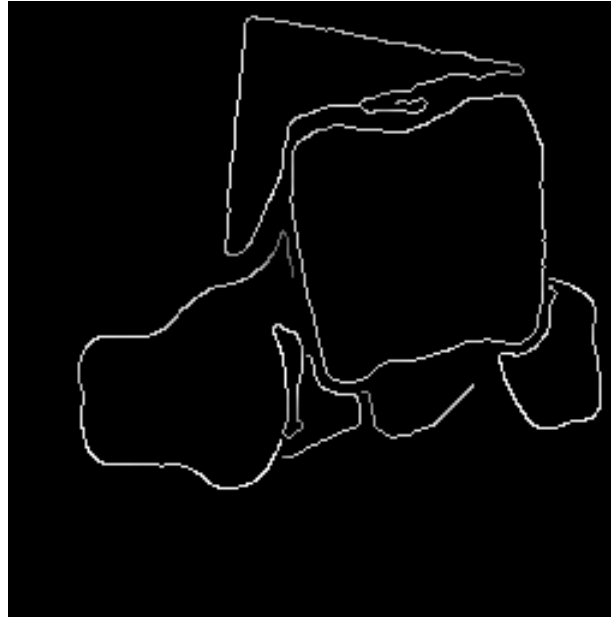


# Canny



Gausov kernel sa  $\sigma = 1.8$ ,  $T_1 = 255$ ,  $T_2 = 1$

Neke ivice koje su nastale usled promene orijentacije površine su ostale, ali su slabije od okvira – regulišemo pragovima



Skaliranje prethodne slike za 0.25 i onda primena Canny sa  $\sigma = 1.8$ ,  $T_1 = 200$ ,  $T_2 = 1$

# Biranje operatora

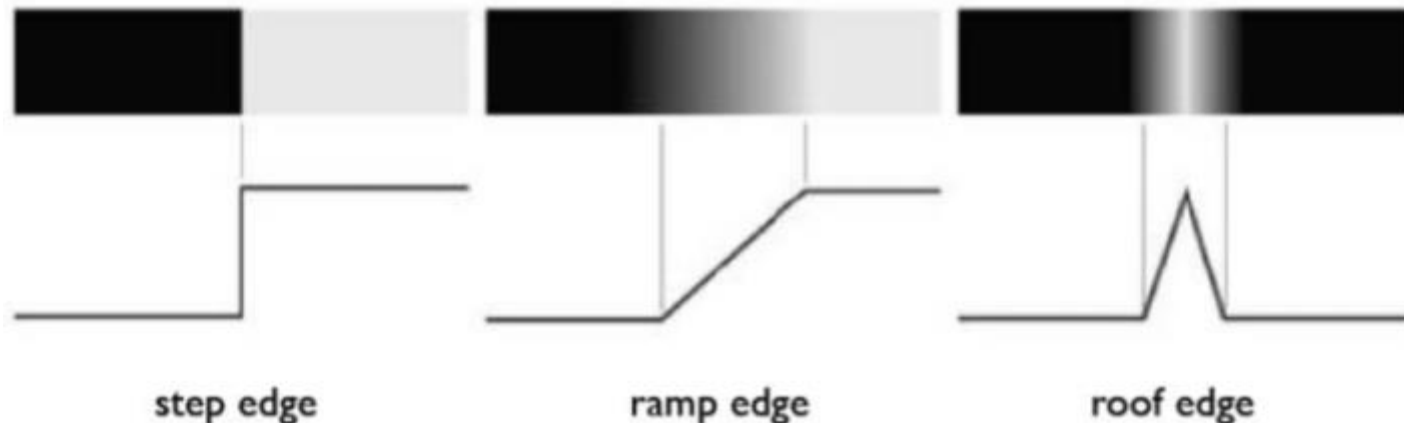
---

- Kao što smo videli, postoji veliki broj konvolutivnih filtera koji omogućavaju detekciju ivica
  - Ideja je da konstruišemo filter koji je osetljiv na velike gradijente na slici, a vraća nule u uniformnim regijama
- Orijentacija ivica
  - Geometrija operatora određuje karakterističan pravac ivica na koje je operator posebno osetljiv
  - Operatori se mogu optimizovati da traže horizontalne, vertikalne ili dijagonalne linije
- Šum
  - Detekcija ivica je teška na slikama sa šumom budući da i objekat na slici i šum stvaraju diskontinuitet na slici
  - Pokušaju da se šum smanji uglavnom rezultuju zamućenim ivicama
  - Operatori koji se koriste na slikama sa šumom su tipično veći (uprosečavanjem više podataka smanjuju uticaj piksela šuma). Nedostatak ovog pristupa je manje tačna lokalizacija detektovanih ivica

# Biranje operatora

- Struktura ivica

- Nemaju sve ivice nagao prelaz intenziteta
- Npr. loš fokus može izazvati postepen prelaz intenziteta i prema tome moramo birati operator
- Postoje novije *wavelet-based* tehnike koje su u mogućnosti da okarakterišu prirodu tranzicije za svaku ivicu
  - Npr. mogu da razlikuju ivice povezane sa kosom od ivica povezanih sa licem



# Zero-crossing operator

