

# Soft kompjuting

Hough transformacija

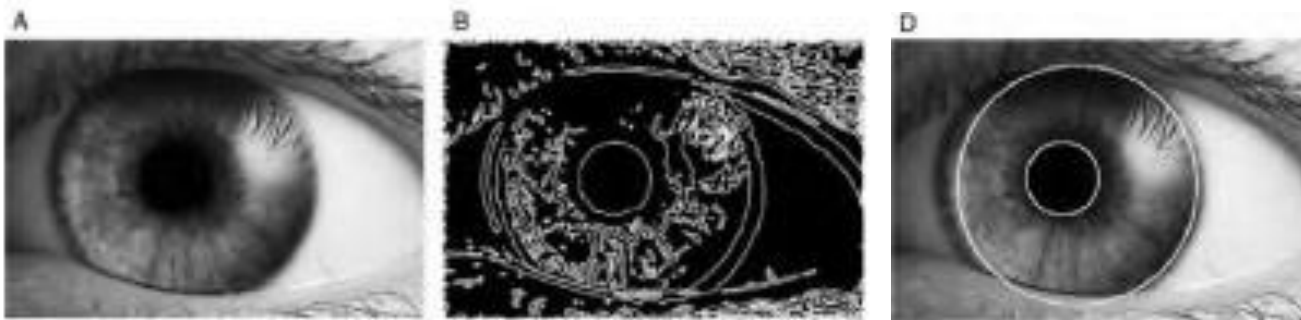
# Reprezentacija znanja

---

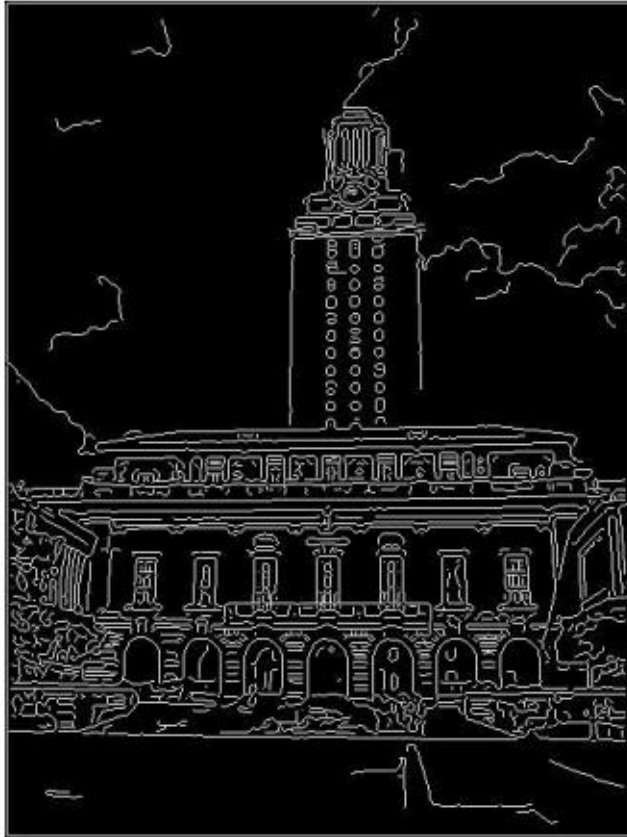
- Kada vršimo analizu slike (npr. detekciju objekata), način na koji reprezentujemo znanje o slici je veoma važan
  - Originalnu reprezentaciju (sliku) transformisaćemo u drugu reprezentaciju koja će nam olakšati rešavanje nekih problema
  - Želimo da u značajnoj meri smanjimo količinu podataka o slici, a da očuvamo važne strukturne informacije
  - Nije uvek trivijalno

# Detekcija ivica

- Detekcija ivica nam omogućava da u značajnoj meri smanjimo količinu podataka na slici
  - Međutim, izlaz operatora za detekciju ivica je i dalje slika (opisana pikselima)
  - Količina informacija bi se još više smanjila ako bismo te piksele pretvorili u jednačine osnovnih geometrijskih oblika
- Nije jednostavno...
  - Zbog nesavršenosti, bilo u podacima ili u detektoru ivica, može biti nedostajućih piksela na željenim krivama
  - Takođe može biti odstupanja od idealnih linija/krugova/elipsi
  - Može biti šuma



# Teškoće pri fitovanju linije



- Suvišne tačke, više modela
  - Koje tačke idu sa kojim linijama?
  - Da li uopšte pripadaju liniji?
- Na nekim linijama postoje procepi
- Šum u izmerenim gradijentima i orijentacijama
  - Kako detektovati stvarne parametre?



# Hough transformacija

---

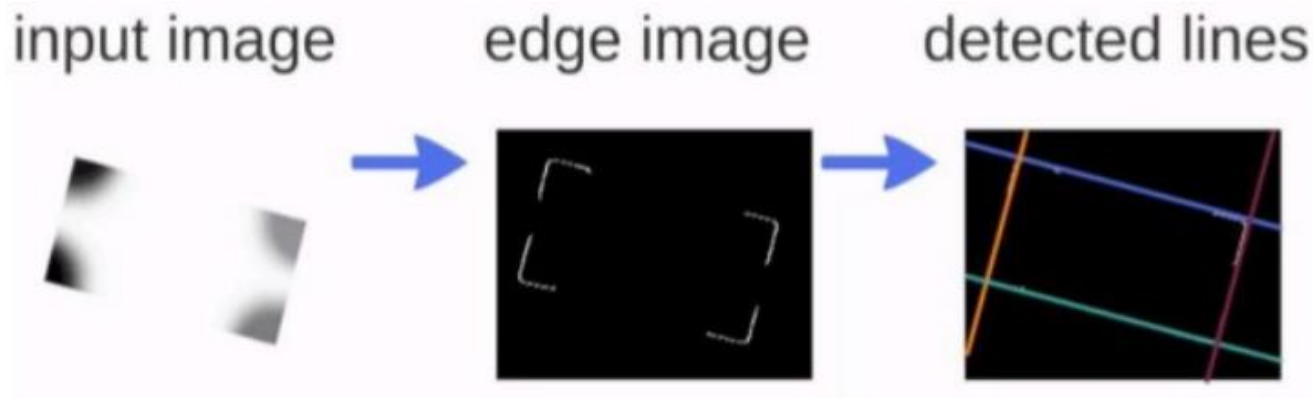
- Svrha *Hough* transformacije je da adresira ovaj problem grupisanja tačaka ivice u objekte
- Možemo je koristiti za detekciju proizvoljnih oblika – ako možemo taj oblik da predstavimo u matematičkoj formi
  - Detekcija linija
  - Detekcija krugova
  - Ostale parametarske krive (proizvoljni oblici)
- Robustna je na šum i delimično zaklanjanje (*partial occlusion*) i donekle na *distortion*
- Koristi se kao tehnika u analizi slike, digitalnoj obradi slika i *computer vision*

# Hough transformacija

---

- Prvi korak (pretprocesiranje) jeste neki algoritam za detekciju ivica
  - Robert Cross (<http://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm>)
  - Sobel (<http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm>)
  - Canny (<http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm>)
  - ...

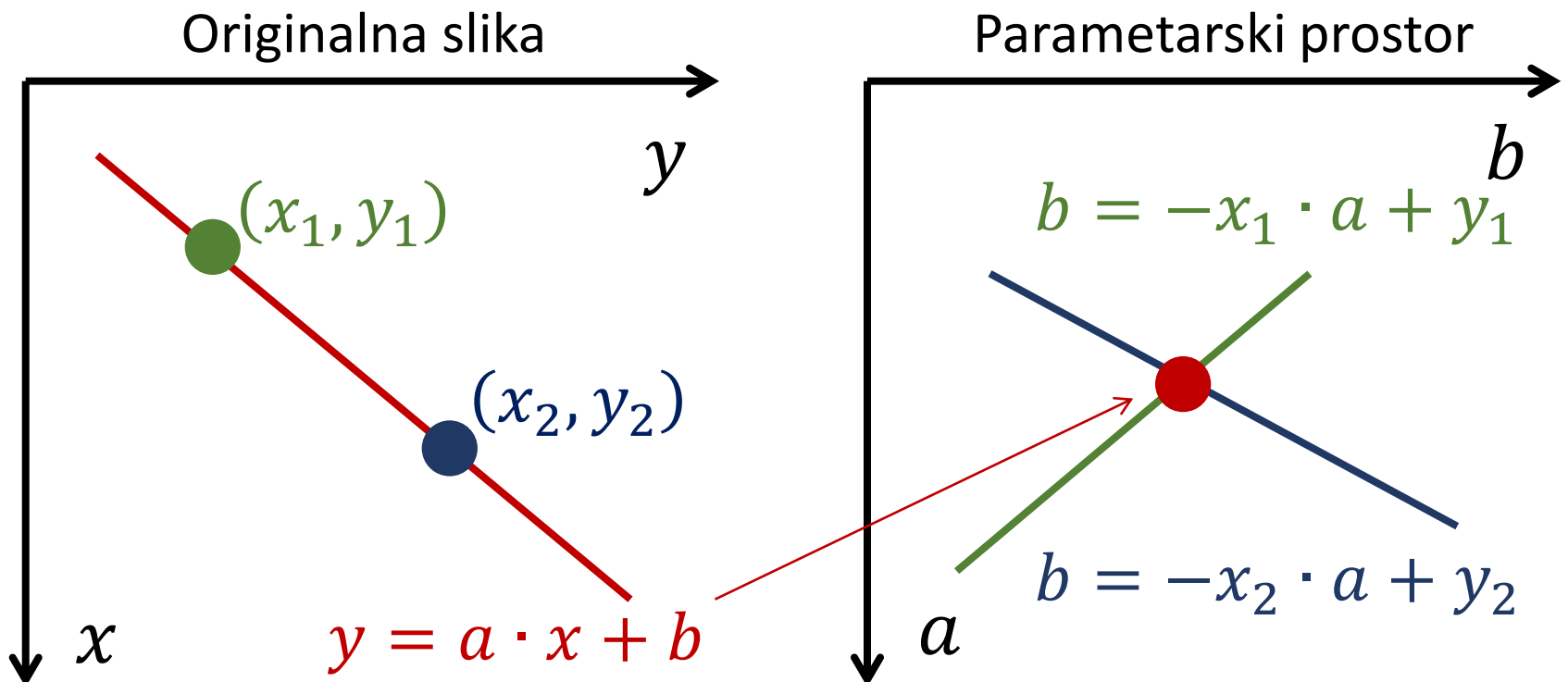
# Hough transformacija



- Rezultati detekcije ivica nam ne govore ništa o identitetu i broju obeležja koje smo pronašli na slici
- Rezultati sadrže šum – imamo više piksela za svako obeležje i procepe
- Ako primenimo *Hough* transformaciju, ovde možemo detektovati 4 linije što nam govori o stvarnoj geometrijskoj strukturi
- *Hough* transformacije kao ulaz uzima binarnu sliku i transformiše je u **parametarski prostor** (*feature space*)

# Detekcija linije

- Svaka linija se može predstaviti jednačinom  $y = a \cdot x + b$
- Dakle, dva parametra  $a$  i  $b$  jedinstveno određuju pravu
- *Hough* prostor ćemo definisati kao parametarski prostor – svih mogućih nagiba  $a$  i preseka sa  $y$ -osom  $b$





# Detekcija linije

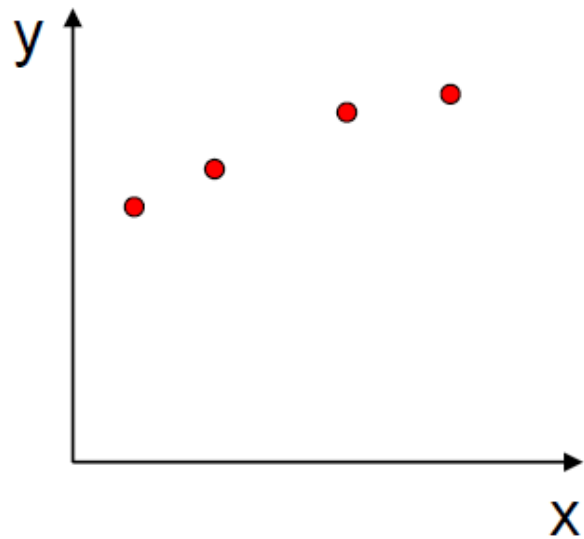
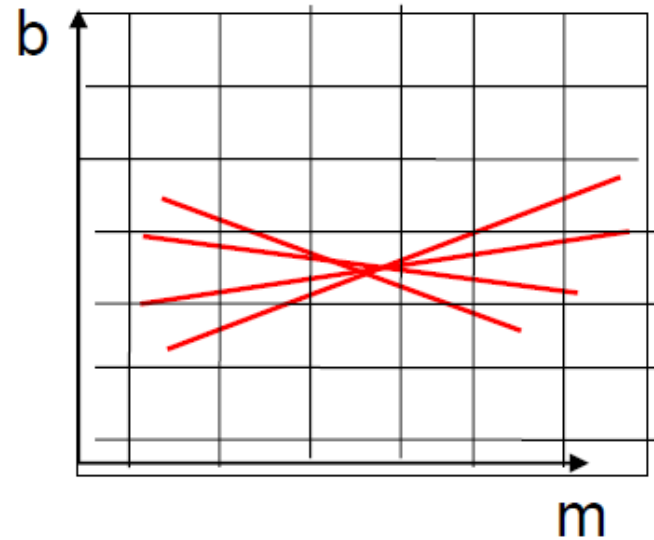


image space



Hough (parameter) space

- Svaka tačka ivice će glasati za skup mogućih parametara u *Hough* (parametarskom) prostoru
- Pikseli koji predstavljaju šum će takođe glasati, ali, tipično, njihovi glasovi će biti nekonzistentni sa većinom „dobrih“ glasova
- Glasove akumuliramo u diskretan broj ćelija (*bins*). Parametri sa najviše glasova predstavljaju linije na slici (*image space*)

# Nedostatak reprezentacije $y = a \cdot x + b$

---

- Nedostatak prethodnog pristupa su vertikalne linije

- Npr. imamo dve tačke (3,1) i (3,2):

$$b = -x \cdot a + y$$

$$(3,1) \Rightarrow b = -3a + 1$$

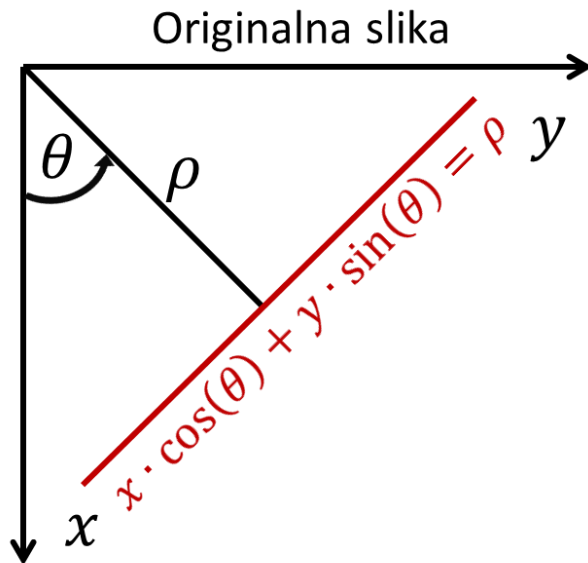
$$(3,2) \Rightarrow b = -3a + 2$$

- U parametarskom prostoru, ove dve prave su paralelne i ne možemo naći tačku preseka koja bi nam dala parametre  $a$  i  $b$
- Rešenje: koristiti drugačiju reprezentaciju prave

# Detekcija linije

- Jednačina prave:  $x \cdot \cos(\theta) + y \cdot \sin(\theta) = \rho$
- Ova jednačina se može predstaviti u obliku sličnom prvoj reprezentaciji:

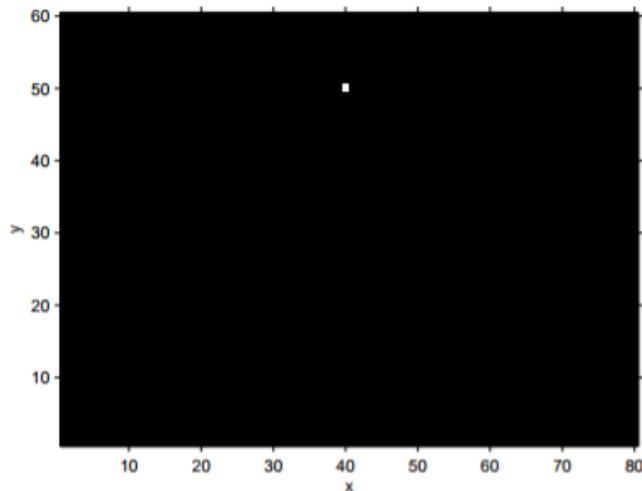
$$y = -\frac{\cos(\theta)}{\sin(\theta)} \cdot x + \frac{\rho}{\sin(\theta)}$$



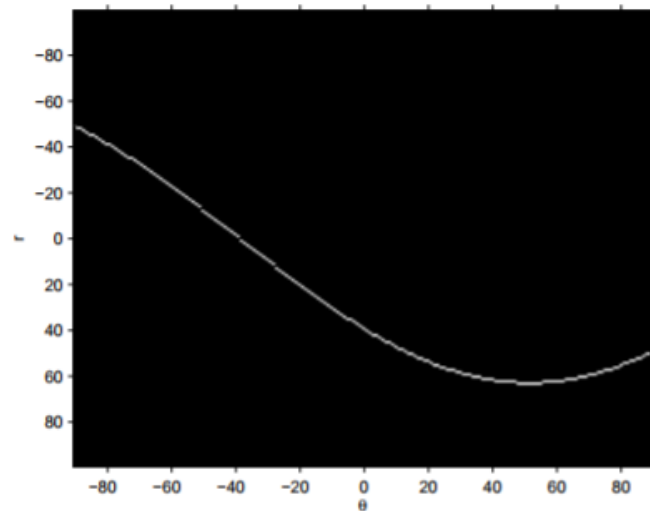
- $\rho$  – rastojanje od koordinatnog početka do najbliže tačke linije
- $\theta$  – ugao  $x$ -ose i linije koja povezuje koordinatni početak sa najbližom tačkom linije

# Hough prostor za skup pravih linija

- Dakle, svaku pravu možemo predstaviti kao funkciju dva parametra –  $\theta$  i  $\rho$
- Parametarski prostor se u ovom slučaju često zove *Hough* prostor za skup pravih linija u dve dimenzije
  - I u ovom slučaju piksel (tačka) se mapira na sve linije koje kroz njega prolaze
  - U *Hough* prostoru će linije izgledati slično sinusoidama

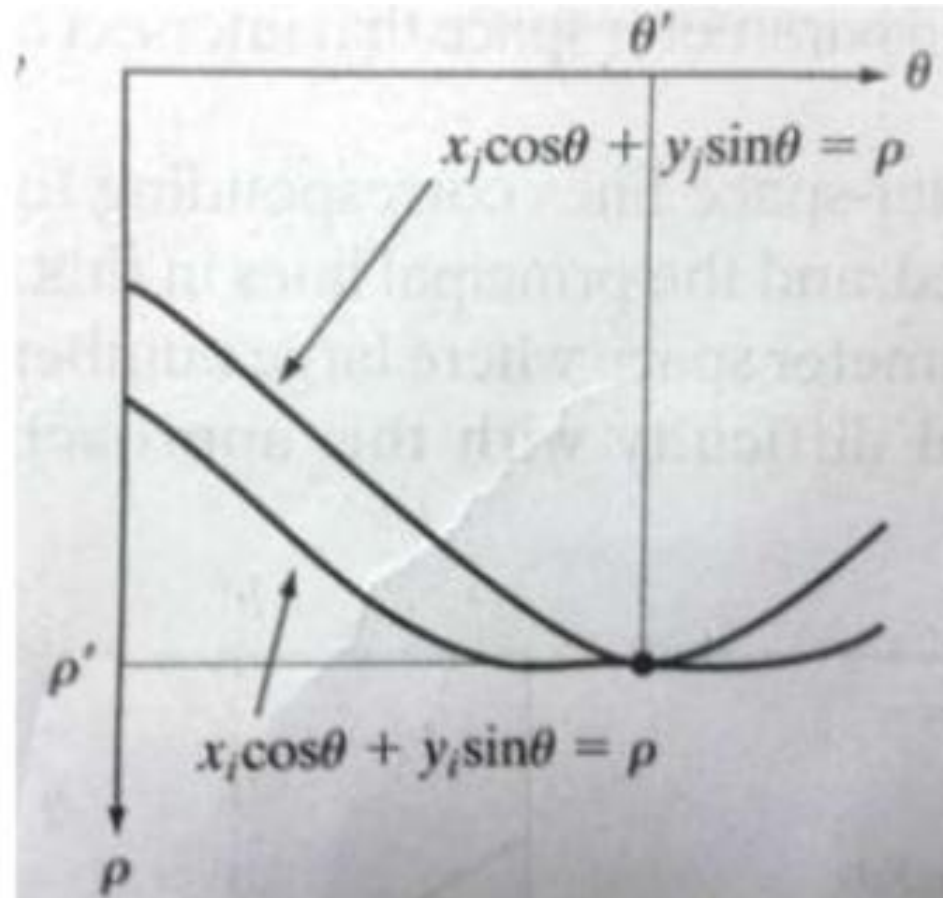
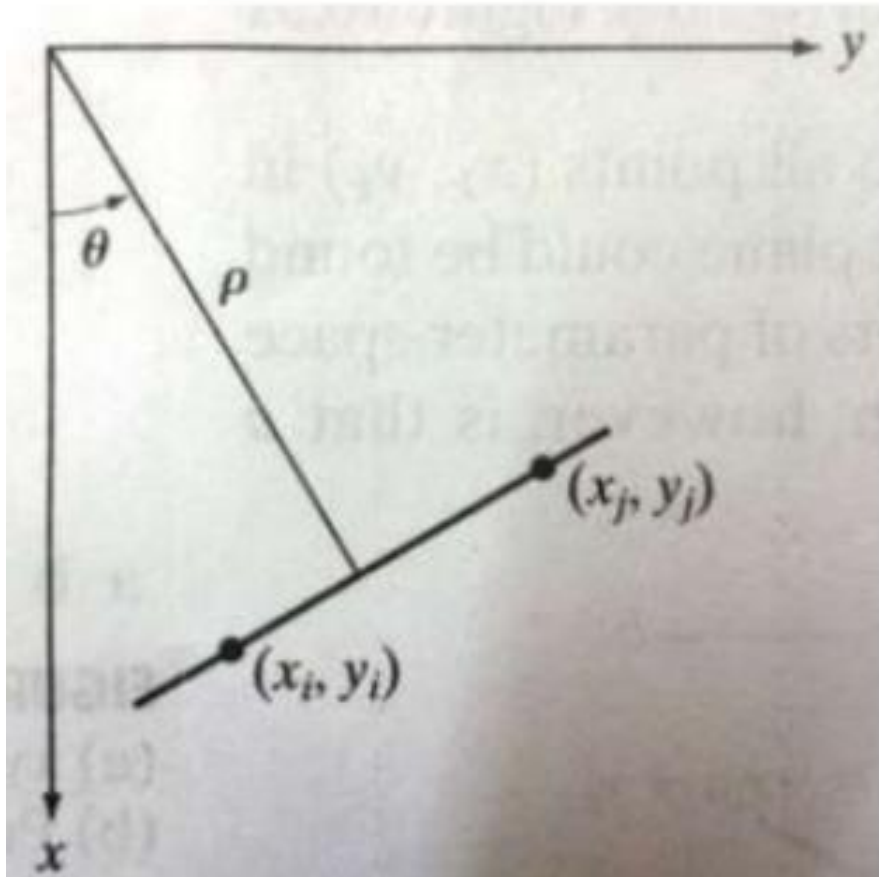


(a) Point  $p_0$ .



(b) All possible lines through  $p_0$  represented in the Hough space.

# Hough prostor za skup pravih linija

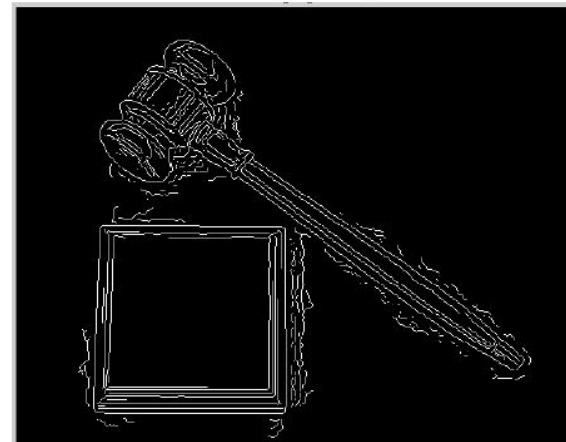


# Hough prostor za skup pravih linija

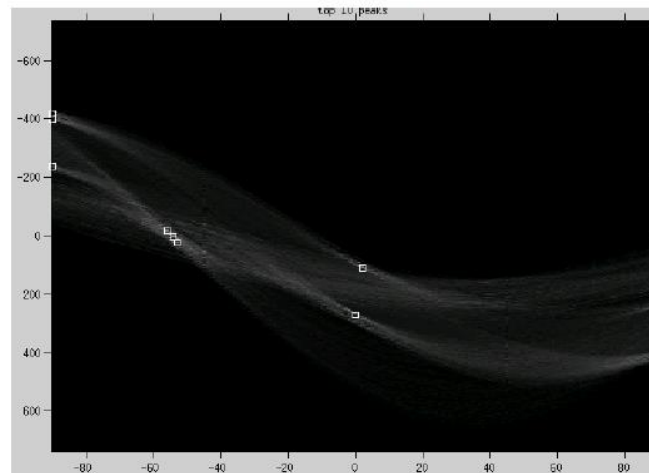
Original image



Canny edges

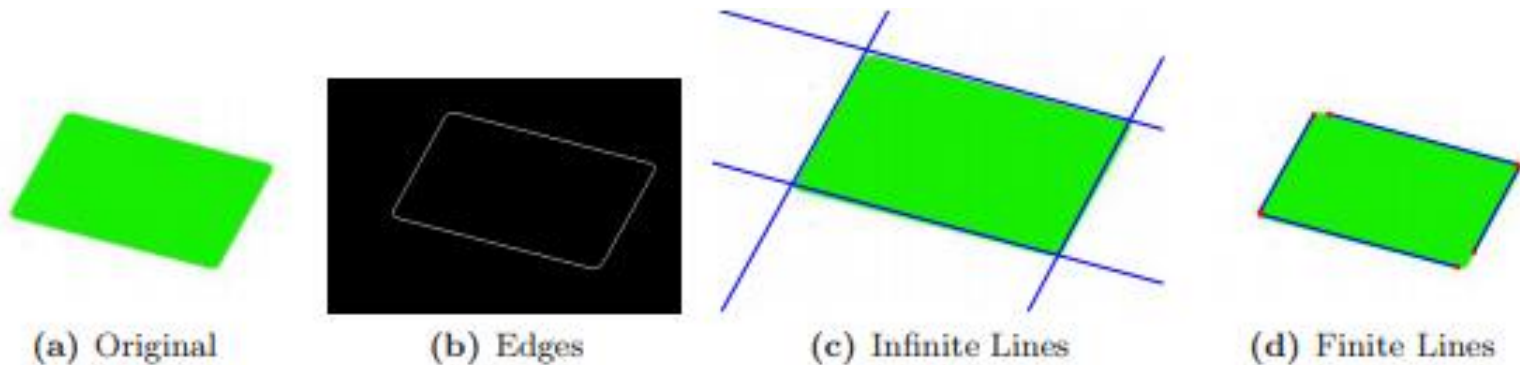


Vote space and top peaks



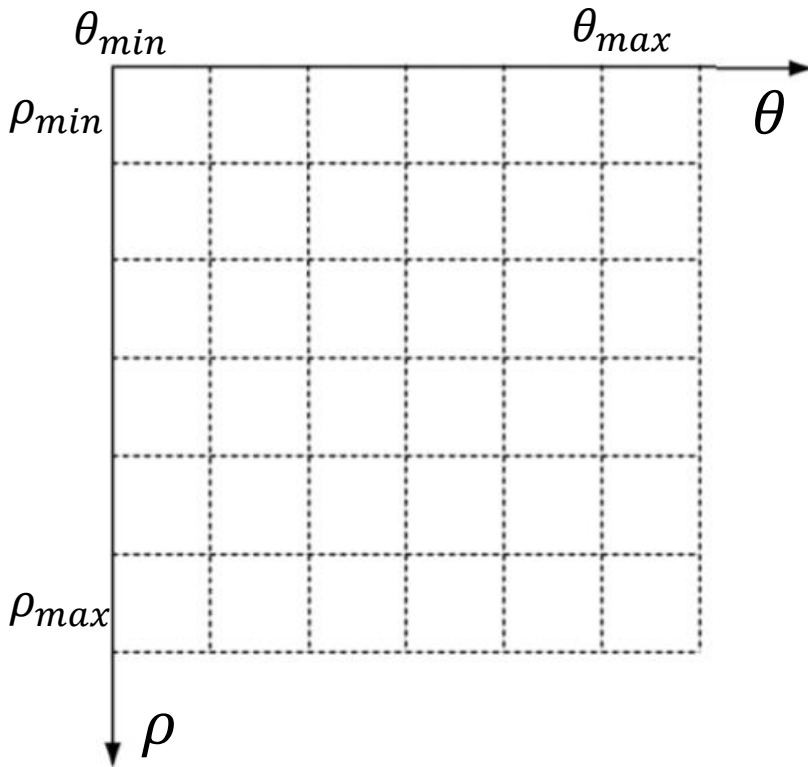
# Algoritam

1. Detekcija ivica
2. Mapiranje piskela sa ivica na *Hough* prostor i snimanje u akumulator (*accumulator*)
3. Interpretacija akumulatora (pronalaženje beskonačnih linija)
4. Konverzija beskonačnih linija u konačne



## 2. Akumulator

- Hough prostor je beskonačan – podelićemo ga u konačne segmente (ćelije)
  - Ovako diskretizovan prostor često zovemo *accumulator cells*



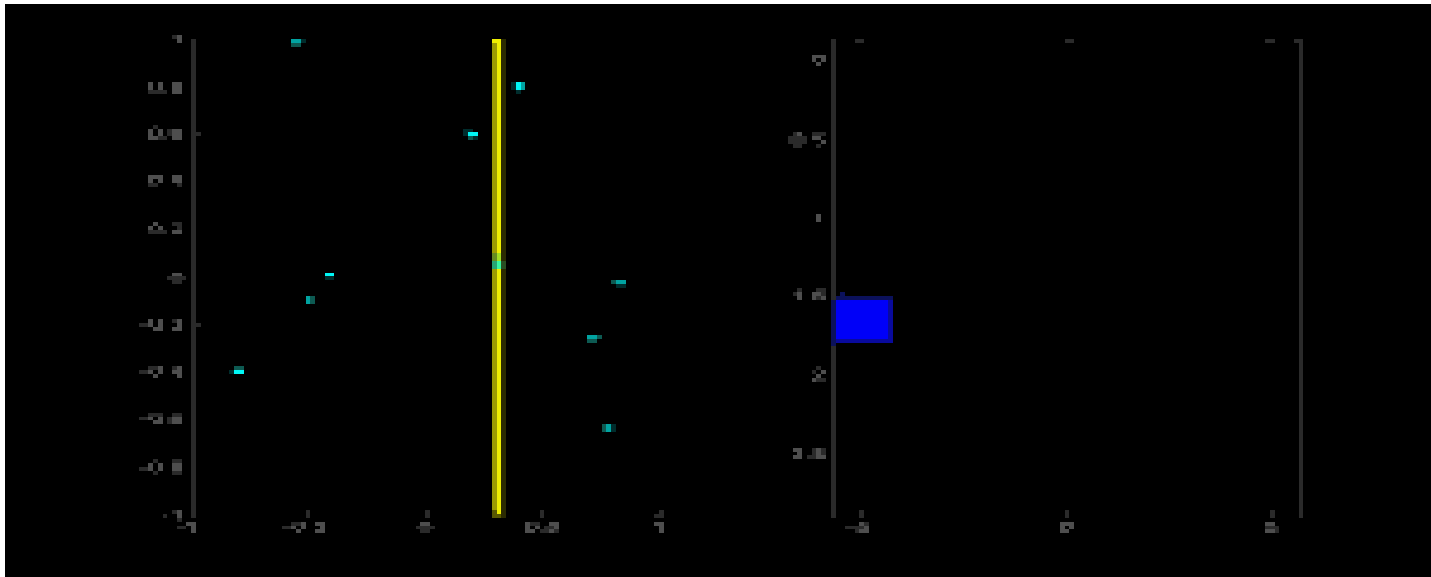
Hough accumulator cells

- U slučaju detekcije linije, akumulator je 2D niz koji čuva vrednosti oba parametra ( $\rho$  i  $\theta$ )
  - U opštem slučaju, dimenzionalnost zavisi od broja nepoznatih parametara
- Vrednost svake ćelije će se uvećati za svaku liniju u *Hough* prostoru koja preseca tu ćeliju



## 2. Akumulator

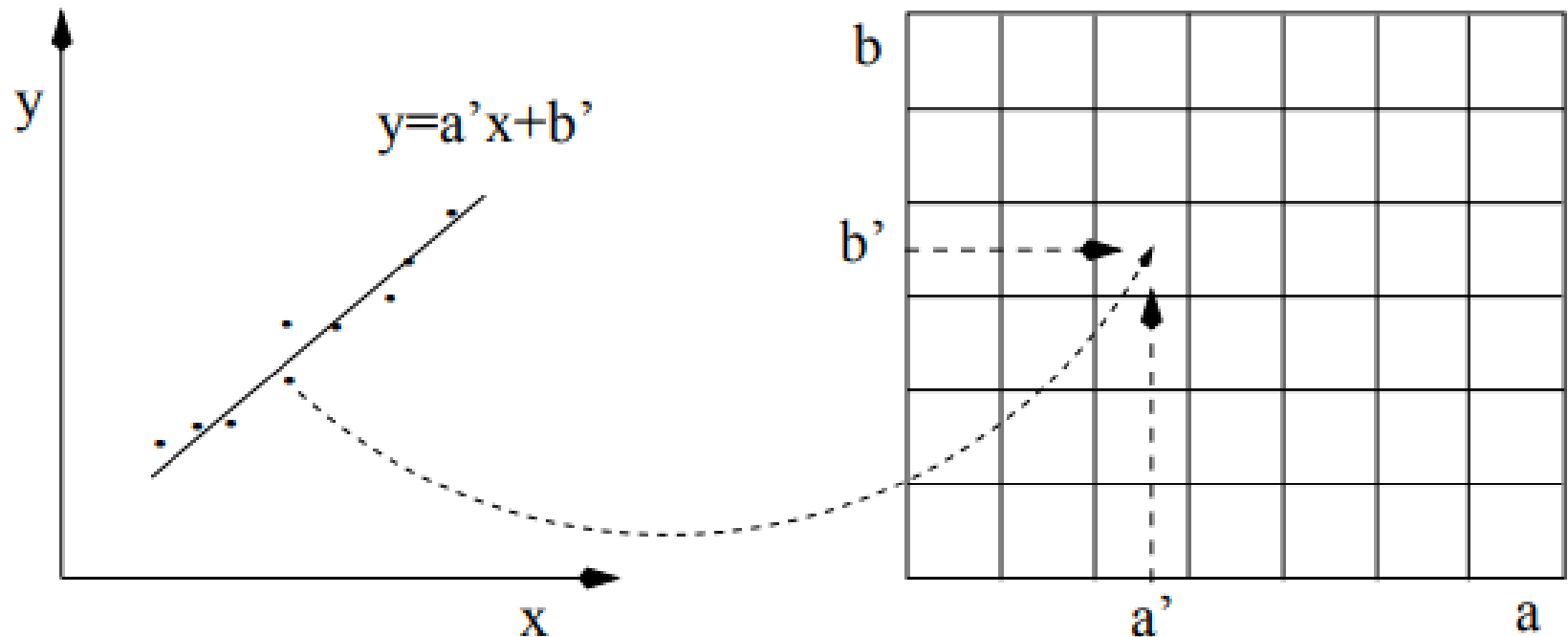
- Za svaki piksel sa vrednošću 1 u binarnoj slici
  - Za svaki ugao  $\theta_j$  iz *diskretnog* niza  $[\theta_{min}, \theta_{max}]$  pronaći  $\rho_j = x \cdot \cos \theta_j + y \cdot \sin \theta_j$
  - Zaokružiti  $\rho_j$  na najbližu ga u najbližu vrednost iz *diskretnog* niza  $[\rho_{min}, \rho_{max}]$
  - Uvećati vrednost odgovarajuće ćelije  $A(\theta_j, \rho_j)$



<http://homepages.inf.ed.ac.uk/amos/hough.html>

## 2. Akumulator – diskretizacija

- „Finija“ diskretizacija:
  - Može tačnije da proceni parametre linije
  - Povećava zahtevano vreme i memoriju
  - U slučaju šuma, „grublja“ diskretizacija je bolja



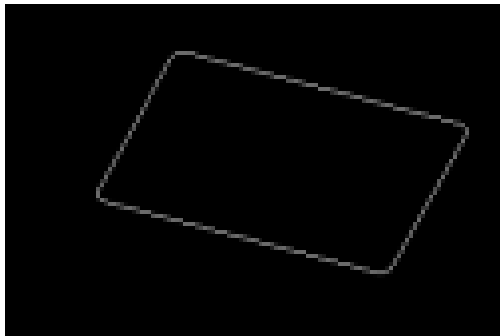
## 2. Akumulator – diskretizacija

---

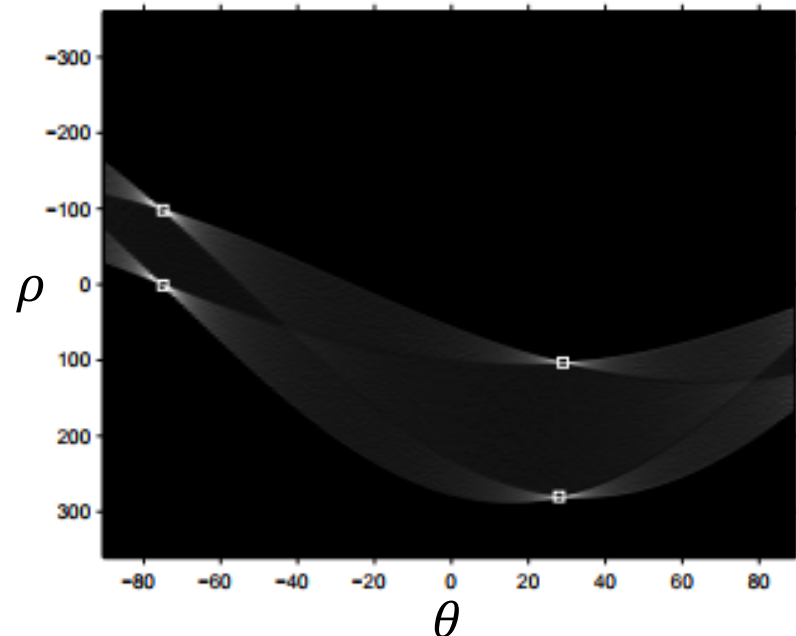
- $\theta \in [-90^\circ, +90^\circ]$ 
  - Horizontalne linije:  $\theta = 0^\circ, \rho \geq 0$
  - Vertikalne linije:  $\theta = 90^\circ, \rho \geq 0$
- $\rho \in [-N\sqrt{2}, N\sqrt{2}]$  za dimenzije slike  $N \times N$ 
  - Maksimalno rastojanje od koordinatnog početka je dijagonala slike
- Dimenzije akumulatora zavise od tačnosti koju želite, npr.
  - ako želite tačnost uglova od  $1^\circ$ , treba vam 180 kolona
  - ako želite preciznost od jednog piksela, broj redova jednak je dužini dijagonale slike

# 3. Interpretacija akumulatora

- Nakon procedure,  $A(i, j) = p$  znači da  $p$  tačaka iz  $(x, y)$  prostora leži na liniji  $\rho_i = x \cdot \cos \theta_j + y \cdot \sin \theta_j$
- Čelije koje imaju najveći broj „glasova“ se smatraju da reprezentuju odgovarajuće linije u  $(x, y)$  prostoru
  - Pronalazimo  $A(i, j)$  kandidate sa vrednošću većom od nekog praga
  - Npr. prag može biti 50% od najveće vrednosti iz akumulatora



(b) Edges

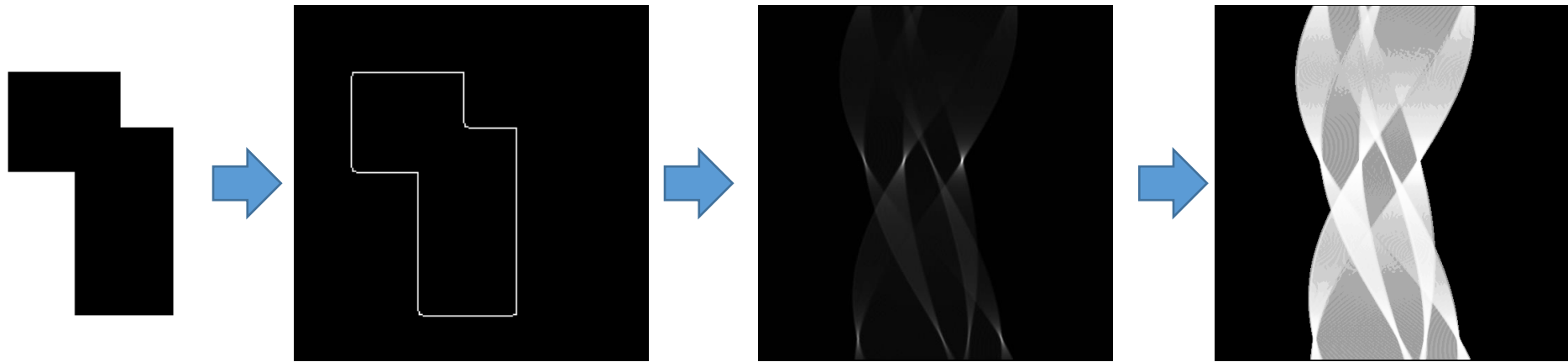


# 3. Interpretacija akumulatora

---

- Pristup sa pragom je nekada dovoljan, ali nije savršen
  - Nekoliko ćelija u akumulatoru koje se nalaze u okolini ćelije koja predstavlja stvarnu liniju mogu imati veliku vrednost
  - Zbog toga pristup sa pragom ima tendenciju da detektuje nekoliko (skoro identičnih) linija za svaku stvarnu liniju
  - Ovo bi se moglo ispraviti definicijom *suppression neighborhood* – dve linije moraju biti značajno različite pre nego što ih obe vratimo kao rezultat

# Napomena



- *Histogram equalizing* nam može pomoći da vidimo šablone tamo gde imamo piksele niskog intenziteta

## 4. Konverzija beskonačnih linija u konačne

---

- *Hough* transformacija vraća parametre  $\rho$  i  $\theta$  i nikakve informacije o dužini linije – sve detektovane linije su beskonačne
- Za linije konačne dužine potrebna je dodatna analiza
  - Za ovo postoji nekoliko algoritama
  - Jedan pristup: čuvamo koordinate svih tačaka u akumulatoru i na osnovu njih limitiramo linije. Mana: mnogo memorije
  - Drugi pristup: vršimo pretragu duž beskonačnih linija na binarnoj slici (sa konturama). Varijanta ovog pristupa je *Probabilistic Hough Transform*

# Progressive Probabilistic Hough Transform (PPHT)

---

- *Hough transform* nije brz algoritam za pronalaženje konačnih linija
  - Dodatno usporeno činjenicom da je analiza za pronalaženje konačnih linija odvojena od samog algoritma
- *PPHT* je način da se ubrza *Hough transform* i simultano detektuju konačne linije
- Ideja:
  - Transformisati slučajno odabrane tačke sa binarne slike
  - Kada određena ćelija akumulatora pređe izvestan broj glasova, binarna slika se pretražuje duž te linije da se vidi da li je jedna ili više konačnih linija prisutno
  - Ako jeste, svi pikseli te linije se uklanjaju sa binarne slike
  - Ceo algoritam:  
[http://web.ipac.caltech.edu/staff/fmasci/home/astro\\_refs/HoughTrans\\_lines\\_09.pdf](http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf)



# *Progressive Probabilistic Hough Transform (PPHT)*

---

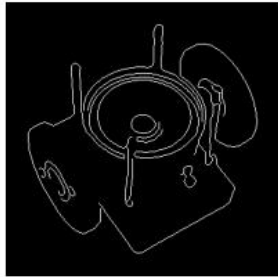
- Nedostaci:

- Više pokretanja algoritma može da vrati različite rezultate – ovo se može desiti ukoliko mnogo linija deli piksele
- Ako se dve linije presecaju, prva detektovana linija uklanja zajedničke piksele stvarajući prazninu u drugoj liniji
- Ako se mnogo linija ukršta, mnogi pikseli mogu nedostajati u poslednjoj liniji i njeni glasovi u akumulatoru mogu da ne dostignu prag potreban za detekciju

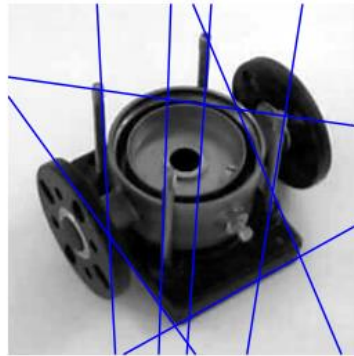
# Primer realne primene *Hough transform*



(a) Original



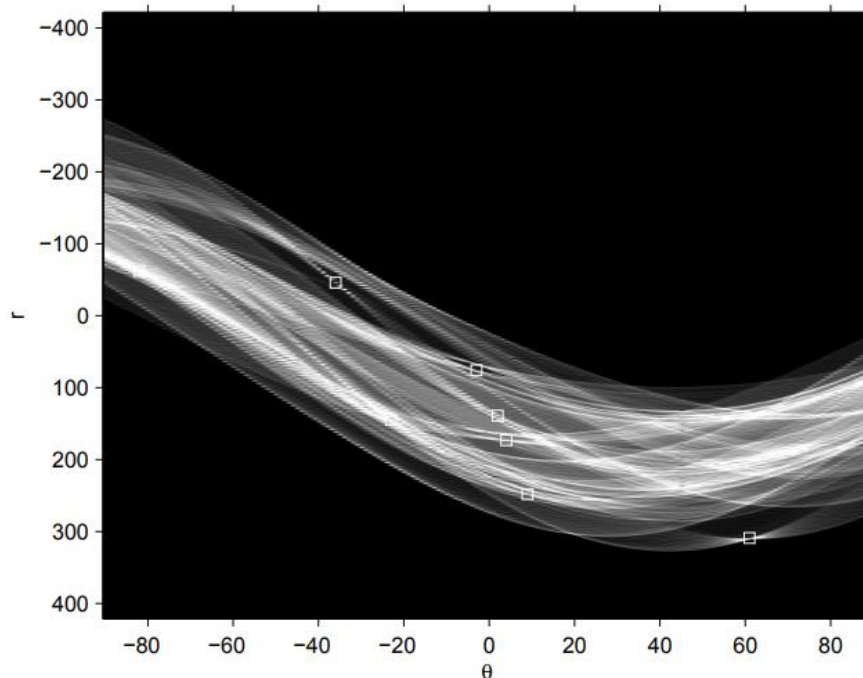
(b) Edges



(c) Infinite Lines



(d) Finite Lines



- Detektovano je 8 ravni linija od kojih su 6 stvarne
- Algoritam je „prevaren“ zbog elipsi na konturama
- Nije moguće neposredno izbeći lažne detekcije a da se očuva većina stvarnih

# U Matlabu

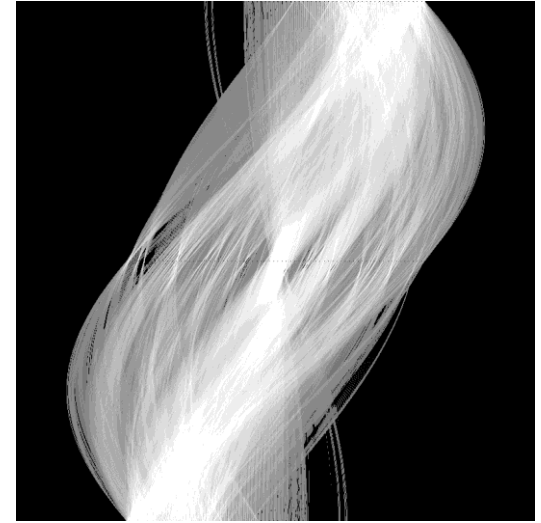
---

- *hough* – vrši *Hough* transformaciju nad binarnom slikom (konturama)
  - U prethodnom primeru rezolucija je 1 za  $\rho$  i  $\theta$
- *houghpeaks* – detektuje linije interpretacijom akumulatora
  - U prethodnom primeru prag je postavljen na 20% maksimalne vrednosti akumulatora, a *supression neighbourhood* na oko 5% rezolucije  $\rho$  i  $\theta$
- *houghlines* – konvertuje beskonačne linije u konačne
  - U prethodnom primeru minimalna dužina linije je 30 piksela
  - Algoritmu je dozvoljeno da poveže linije kroz rupe od maksimalno 30 piksela

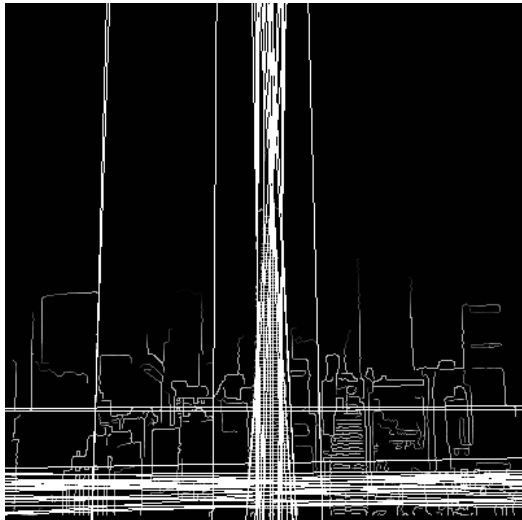
# Primer realne primene *Hough transform*



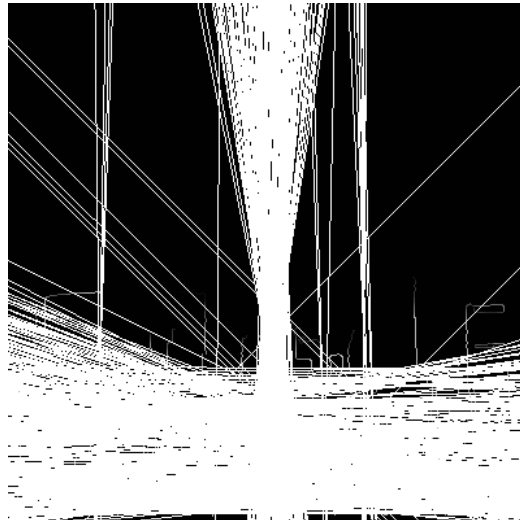
(Canny)



(Histogram equalized)



prag=70%

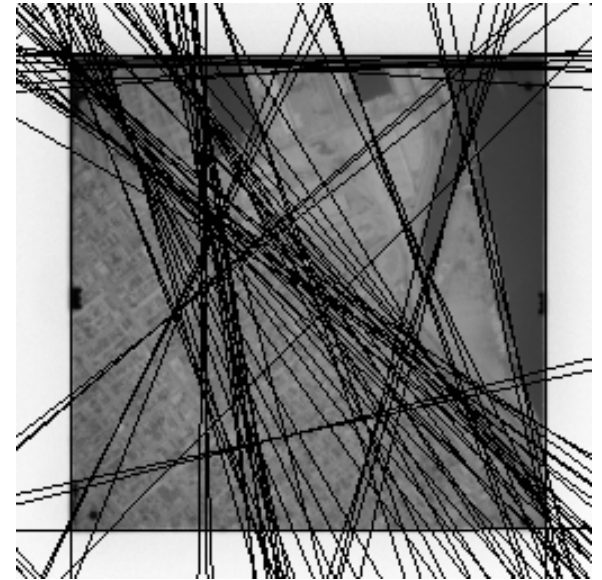


prag=50%

# Primer realne primene *Hough transform*

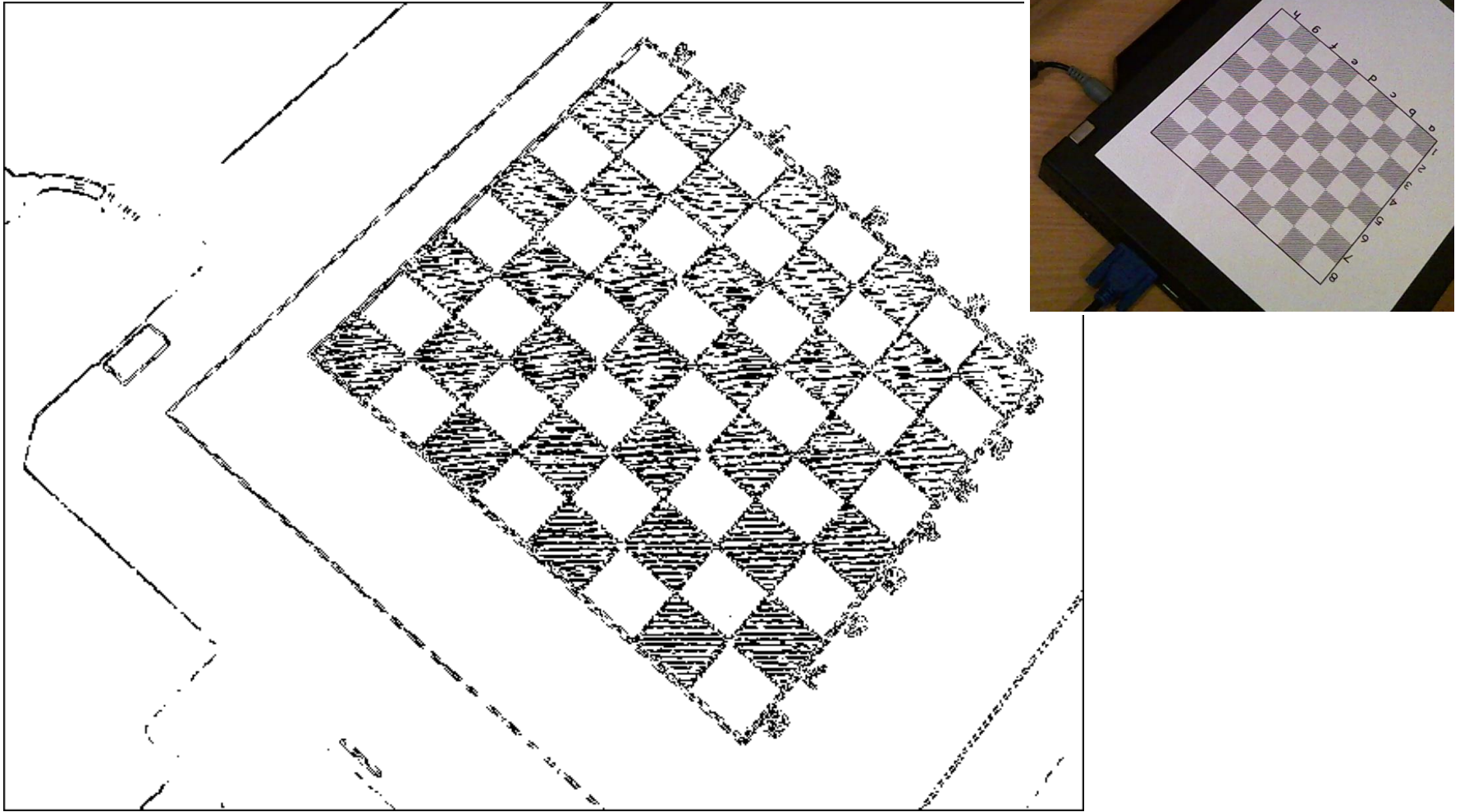


(Canny)



- Pronađeno je nešto informacija o ulicama
- Možda bi pomoglo da pojačamo kontrast originalne slike

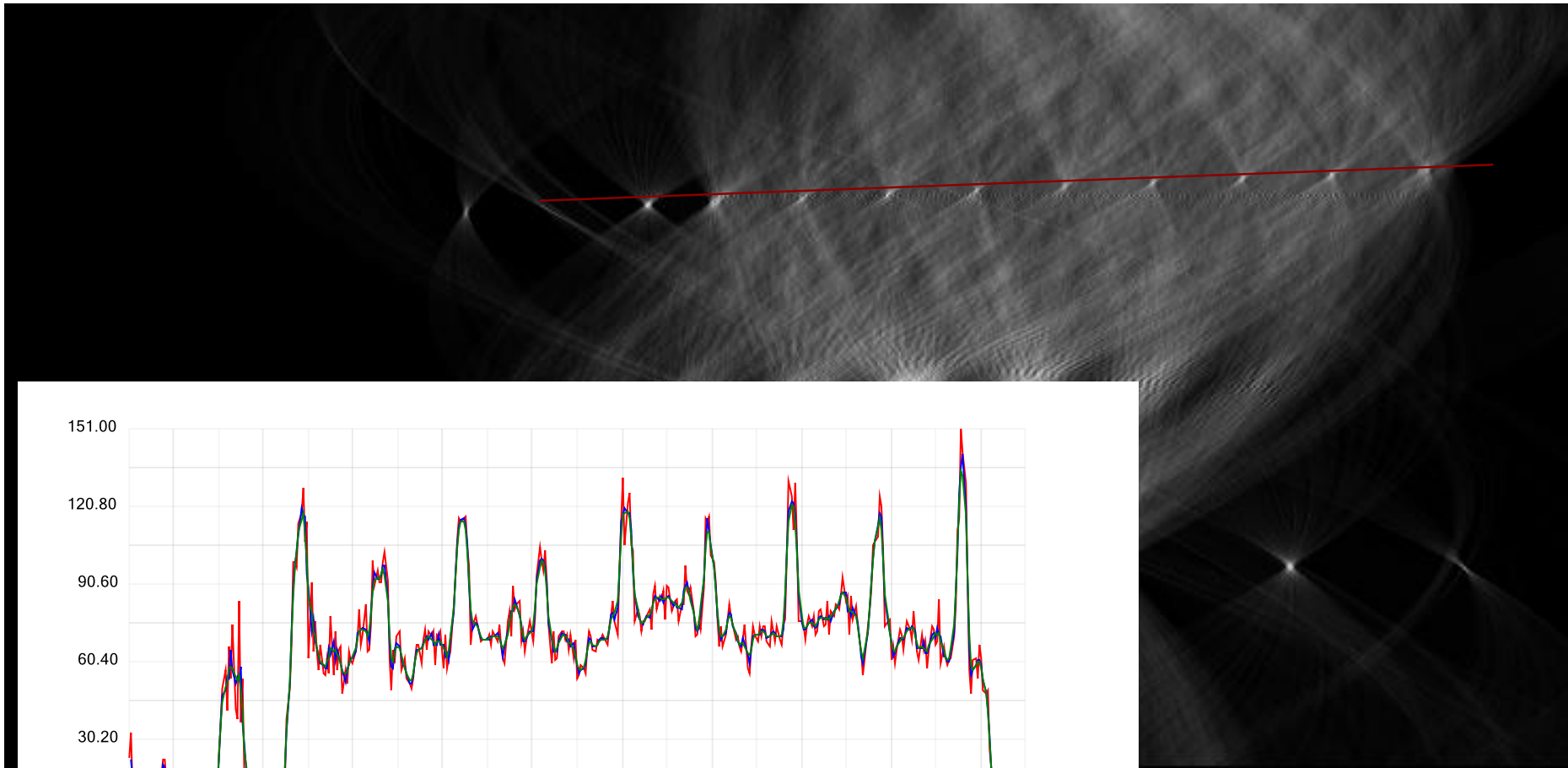
# Primer realne primene *Hough transform*



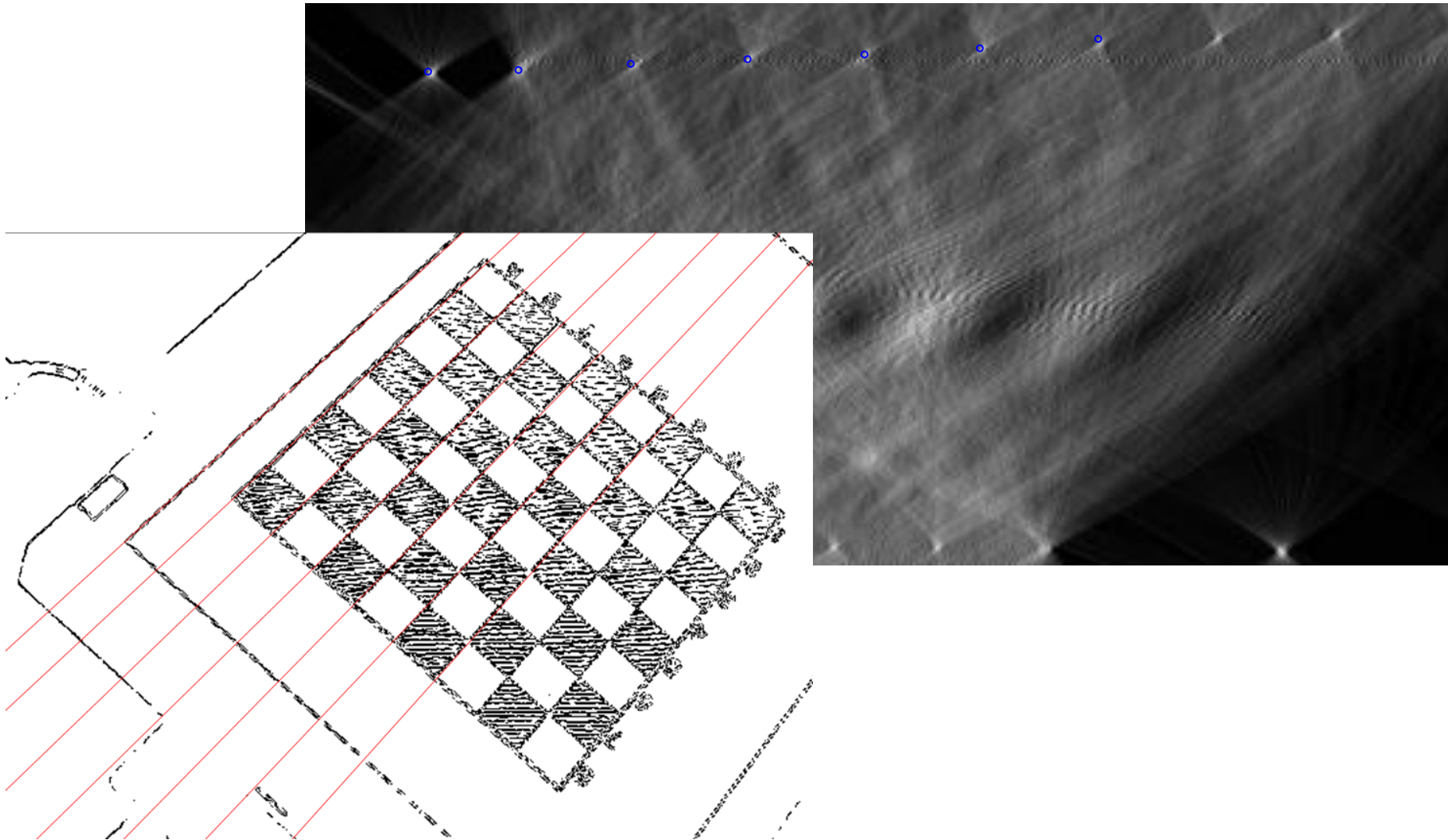
Slika šahovske table nakon izdvajanja ivica i segmentacije.



# Primer realne primene *Hough transform*

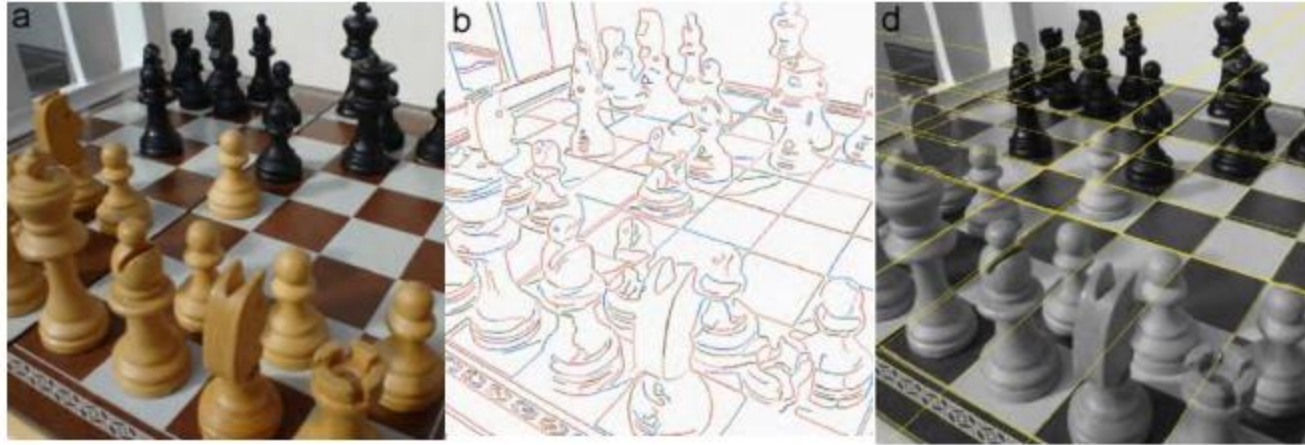


# Primer realne primene *Hough transform*

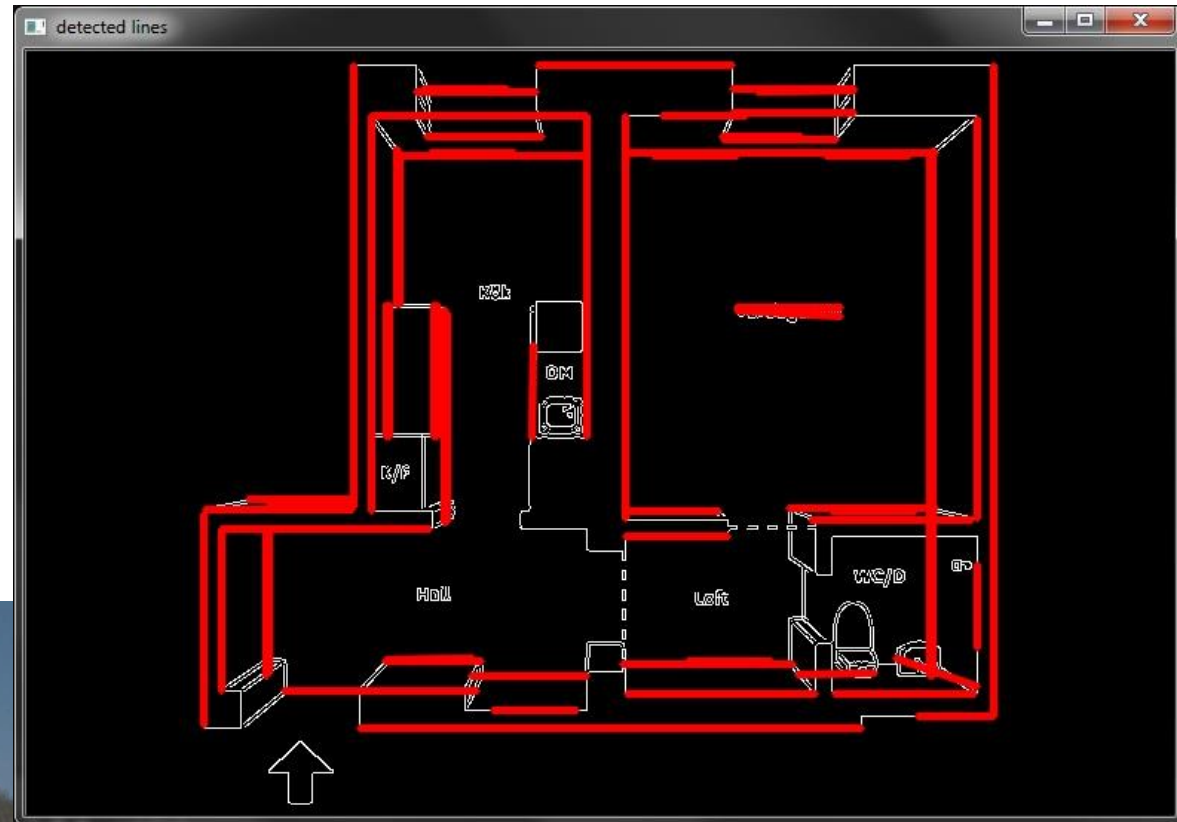
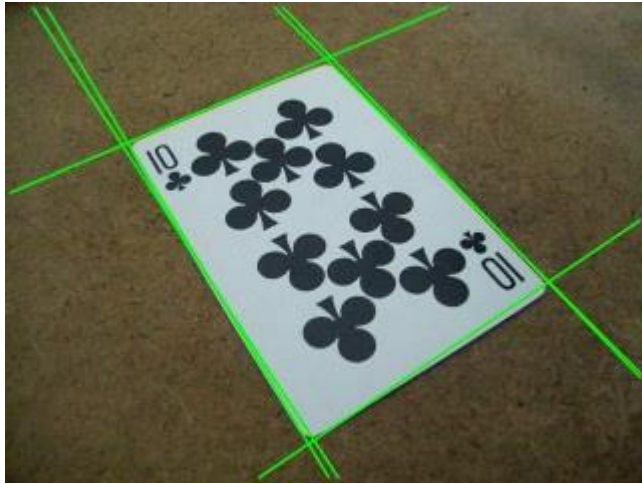




# Primer realne primene *Hough transform*

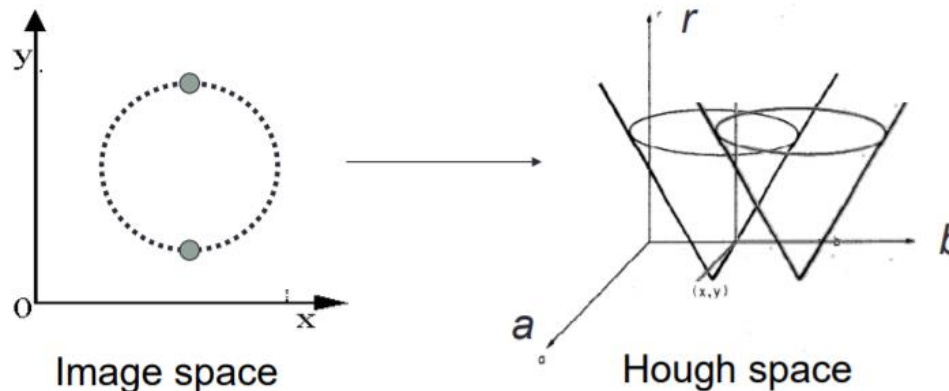


# Primer realne primene *Hough transform*



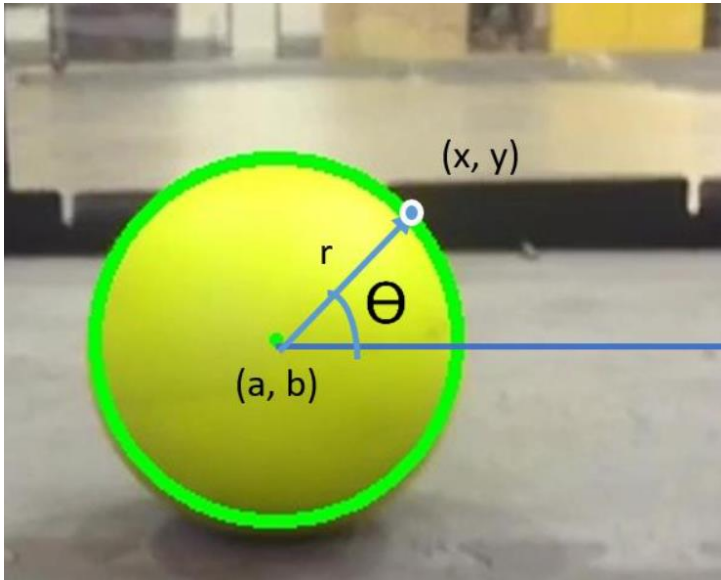
# Proširenje *Hough transform* – krugovi

- Jednačina kruga:  $(x - a)^2 + (y - b)^2 = r^2$
- Dakle, imamo tri parametra:  $(a, b)$  i  $r$  – proširićemo akumulator da bude 3D niz
  - Inicijalizovati sve  $A[a, b, r] = 0$
  - Za svaki piksel  $(x, y)$  binarne slike:
    - Za svako  $(a, b)$ :
      - $r = \sqrt{((x - a)^2 + (y - b)^2)}$
      - $A[a, b, r] = A[a, b, r] + 1$



# Proširenje *Hough transform* – krugovi

- Ovo proširenje je veoma jednostavno za razumevanje i implementaciju, ali i neefikasno
- Krug bismo mogli predstaviti sledećim jednačinama:



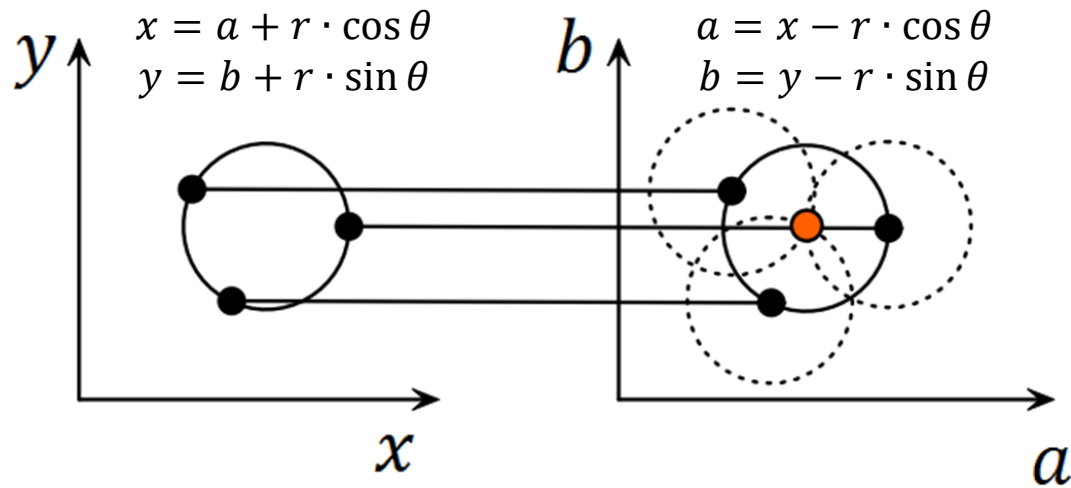
$$x = a + r \cdot \cos \theta$$
$$y = b + r \cdot \sin \theta$$

Ako bi  $r$  bilo poznato, mogli bismo izračunati  $a$  i  $b$ :

$$a = x - r \cdot \cos \theta$$
$$b = y - r \cdot \sin \theta$$

- Svaka tačka u  $(x, y)$  prostoru bi se preslikavala na kružnicu u  $(a, b)$  prostoru, a *Hough* prostor bi bio 2D

# Proširenje *Hough transform* – krugovi



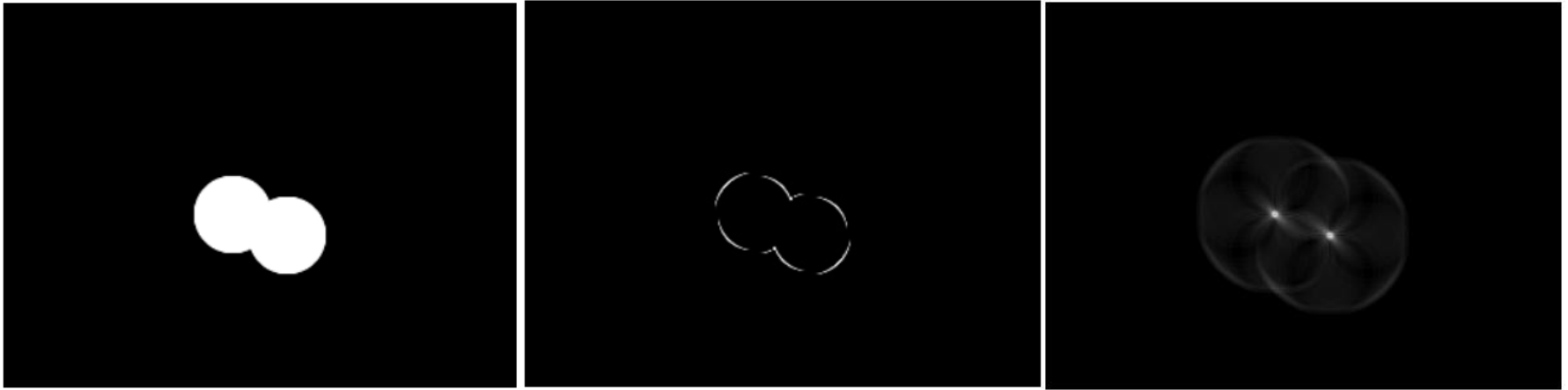
Originalni prostor

- označena su tri piksela na kružnici koju detektujemo

*Hough* prostor (za fiksno  $r$ )

- Svaka tačka se preslikava na kružnicu
- Kružnice predstavljaju centre svih kružnica prečnika  $r$  koji bi mogli da sadrže piksel originalnog prostora

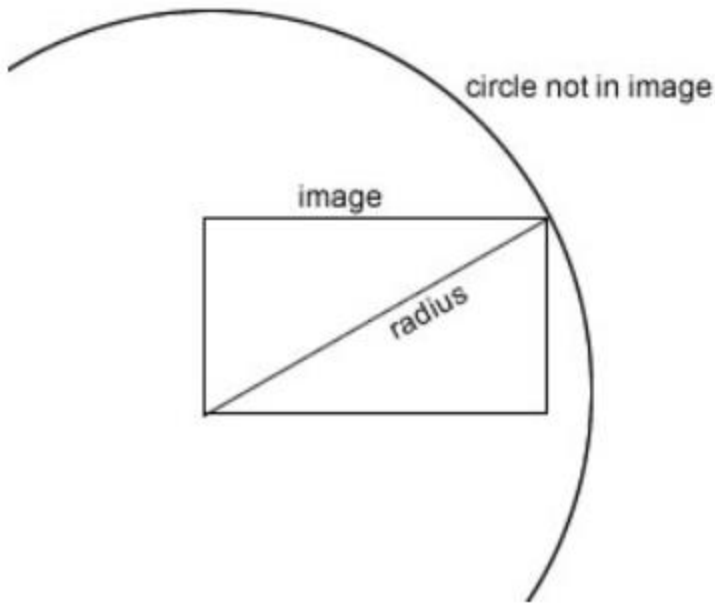
# Proširenje *Hough transform* – krugovi



- Učitamo sliku
- Detektujemo ivice i generišemo binarnu sliku
- Za svaki piksel ivice, generišemo kružnicu u *Hough* prostoru
- U preseku kružnica se nalaze detektovani centri kružnice

# Ali šta kada $r$ nije poznat?

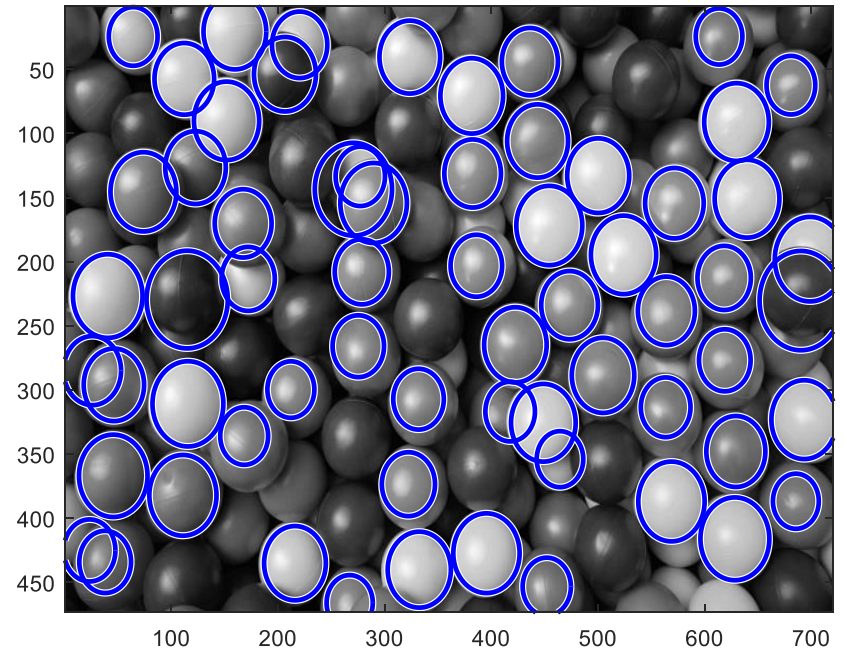
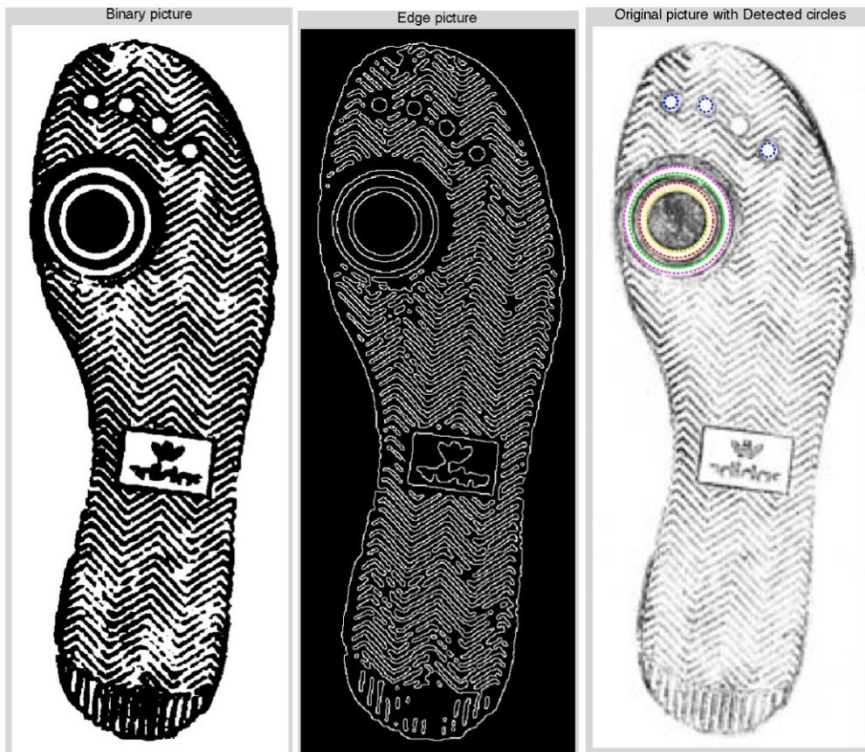
- Pogađamo: probamo isti algoritam za  $r = 1, r = 2, \dots, r = |\text{diagonala\_slike}|$



- Dakle, kontrolisano menjamo  $r$  u nekom rasponu pa je problem sa 3D sveden na 2.5D



# Proširenje *Hough transform* – krugovi



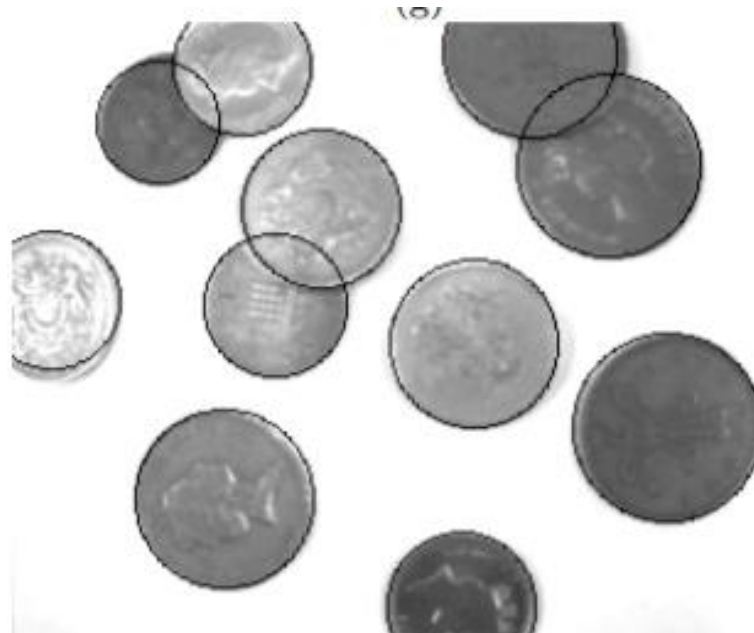


# Proširenje *Hough transform* – krugovi



Detektovanje dužica (iris)

Hough prostor



Radi i ako nije  
vidljiva cela  
kružnica

# Proširenje *Hough transform*

- „Klasičan“ *Hough transform* može da se koristi i za detekciju krugova, elipsi,...

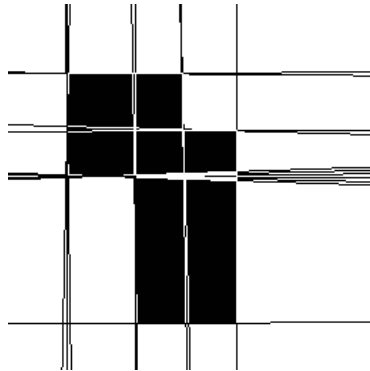
Analytic Form	Parameters	Equation
Line	$\rho, \theta$	$x\cos\theta + y\sin\theta = \rho$
Circle	$x_0, y_0, \rho$	$(x-x_0)^2 + (y-y_0)^2 = \rho^2$
Parabola	$x_0, y_0, \rho, \theta$	$(y-y_0)^2 = 4\rho(x-x_0)$
Ellipse	$x_0, y_0, a, b, \theta$	$(x-x_0)^2/a^2 + (y-y_0)^2/b^2 = 1$

- U opštem slučaju, možemo detektovati bilo koju krivu koju možemo opisati analitički u formi:

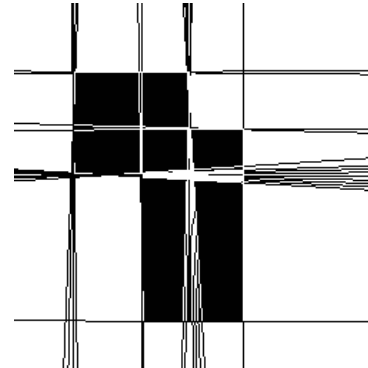
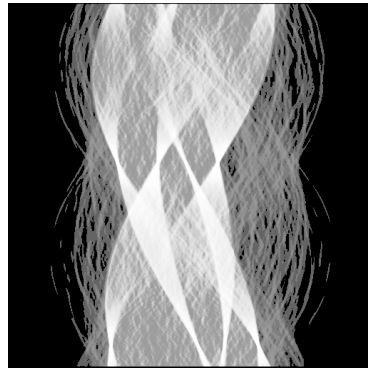
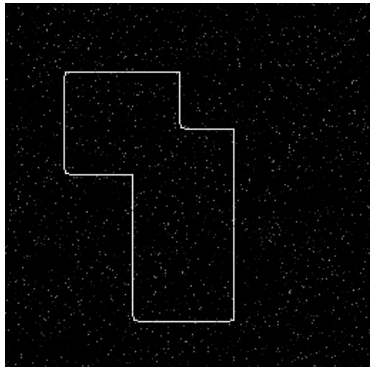
$$g(v, C),$$

gde je  $v$  vektor koordinata, a  $C$  parametri

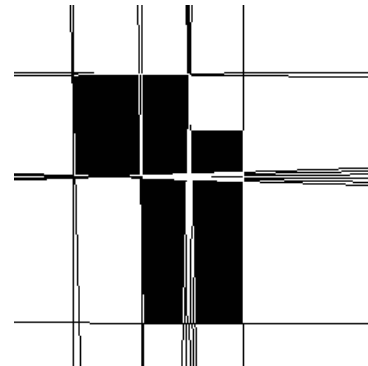
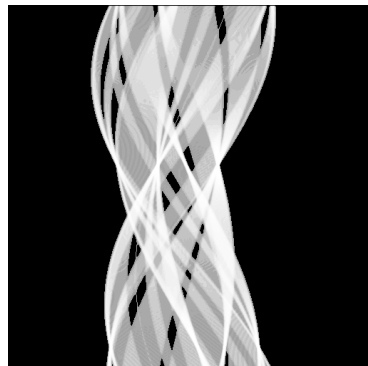
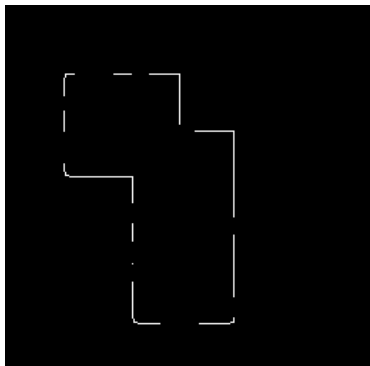
# Šum



- Nije perfektno...
- Možda bi se moglo pobojšati rafiniranjem diskretizacije

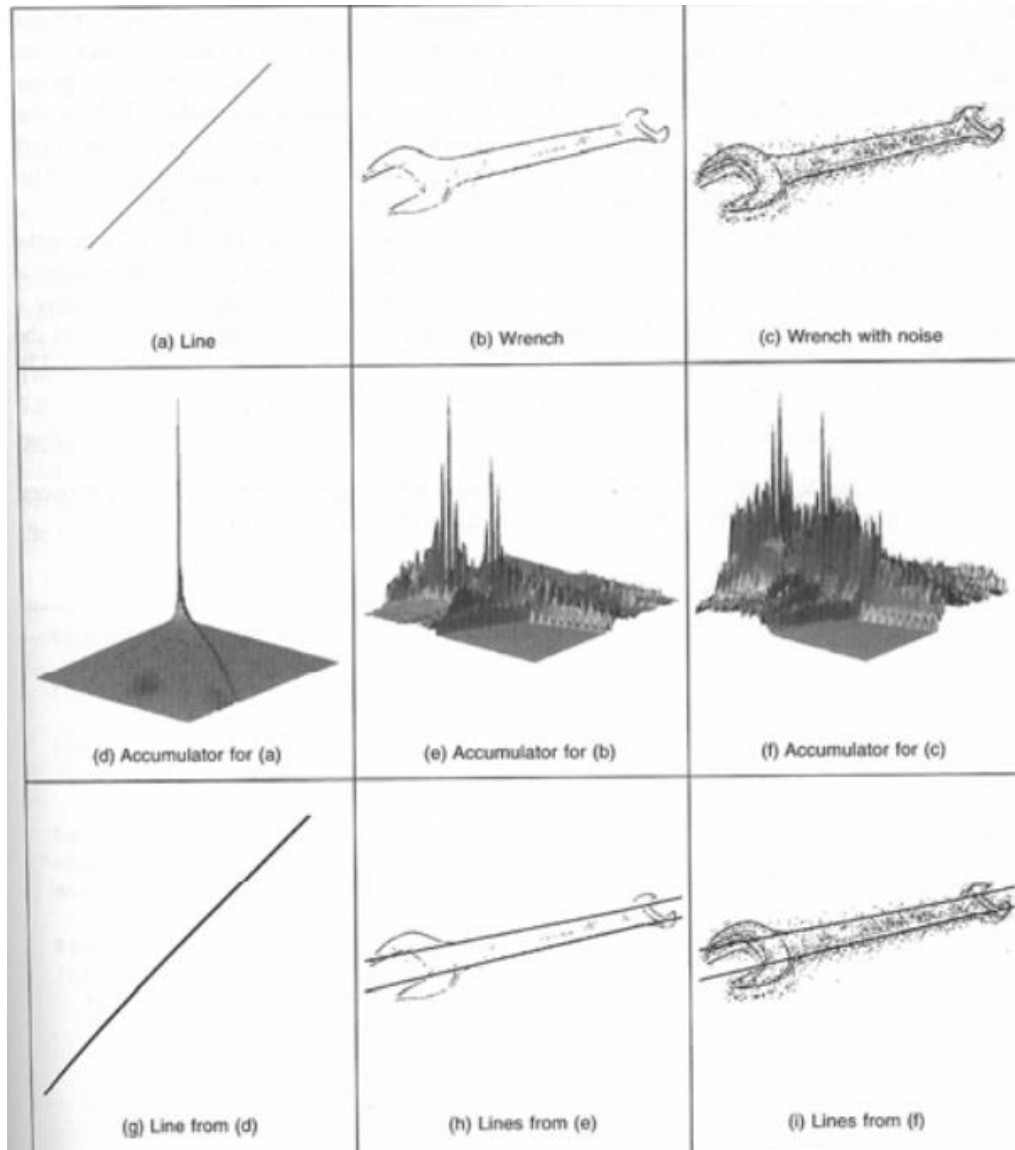


(rezultat je preklopljen sa originalnim)



Samo 7 linija je pronađeno, ali su sve relevantne

# Šum



Binarna slika (ivice)

Vizuelizacija akumulatora  
(visina odgovara broju piksela koji su  
glasali za liniju)

Šum jeste uticao na akumulator, ali i  
dalje najveći vrhovi odgovaraju  
glavnim linijama

Nakon primene praga

# Prednosti *Hough transform*

---

- I pored ograničenja domena, „klasičan“ *Hough transform* je primenljiv u mnogim aplikacijama
  - Npr. mnogi anatomske delovi ispitivani u medicinskim slikama imaju ivice koje mogu da se izraze kao krive
- Glavne prednosti *Hough transform*:
  - Konceptualno jednostavan i lak za implementaciju
  - Tolerancija rupa (*gaps*) u ivicama
  - Tolerancija na šum
  - Može da se adaptira za mnoge oblike (ne samo linije)

# Mane *Hough transform*

---

- Mane:
  - Što više parametara ima kriva, treba nam veća dimenzija akumulatora – *Hough* je zato praktičan samo za jednostavne krive
    - Imajte na umu da treba ispitati svaku moguću kombinaciju vrednosti parametara
  - Traži samo jedan tip objekata
- Moguće je detektovati i proizvoljne oblike (gde ne postoje jednostavan analitički opis): *Generalized Hough Transform*
  - Ovo je takođe veoma računarski zahtevno

# LSD: Line Segment Detector

---

- Pronalazi koherentne linije i krive na slici, umesto da se oslanja na lokalne detektora ivica (gradijent)
- <http://www.ipol.im/pub/art/2012/gjmr-lsd/>
  - Video: možemo videti da su dobijene krive koherentne, ne menjaju se naglo sa promenom frejmova

# LSD: Line Segment Detector

- [http://demo.ipol.im/demo/gjmr\\_line\\_segment\\_detector/](http://demo.ipol.im/demo/gjmr_line_segment_detector/)

