Praktikum zo strojového učenia a umelej inteligencie na vizuálnych dátach

Andrej Lúčny

Katedra aplikovanej informatiky FMFI UK

lucny@fmph.uniba.sk

http://dai.fmph.uniba.sk/w/Andrej_Lucny

www.agentspace.org/praktikum

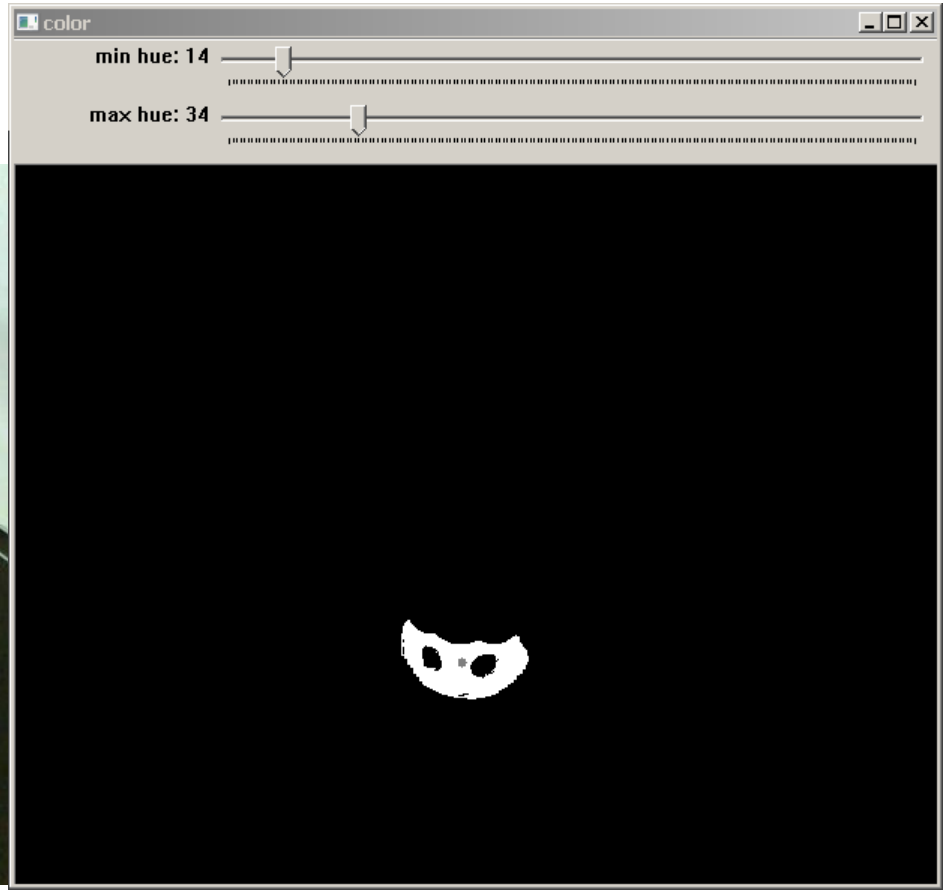
Detekcia objektov

Objekt sa dá detekovať rôznymi stratégiami:

1. Farbou
2. Fázovou koreláciou
3. Rozložením význačných bodov (SIFT, SURF, ORB)
4. Podľa tvaru (Houghova transformácia)
5. Pomocou strojového učenia nad príznakmi (Haar, HOG, LBPH)

Detekcia objektov - farbou

1. Obraz prevedieme do HSV
2. Tým pádom farba je vyjadriteľná intervalom



Det. obj. fázovou koreláciou

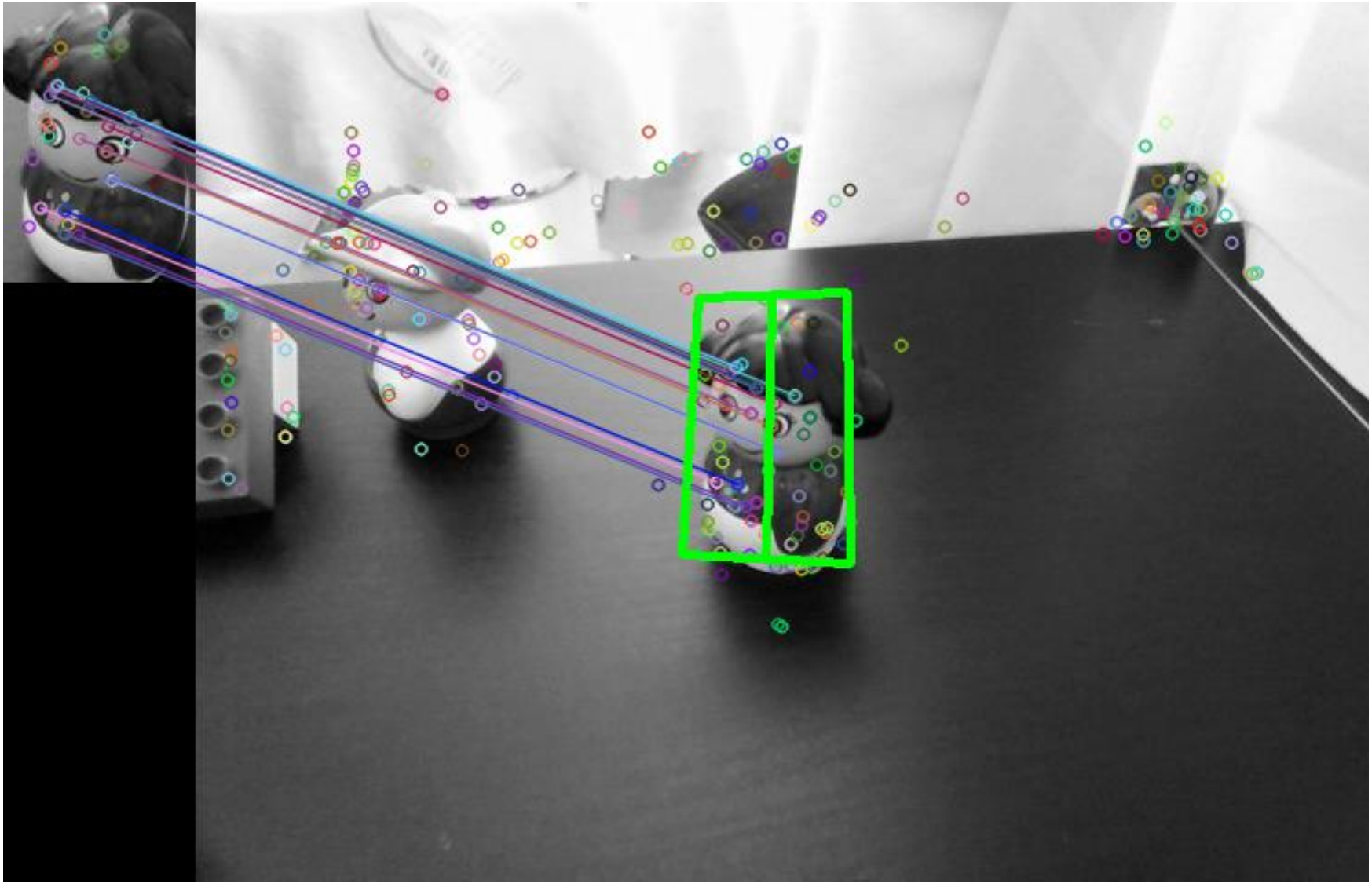
1. Binarizujeme vzor i obraz
2. Vybraný vzor sa snažíme priložiť do každého bodu obrazu, spočítať chybu a vybrať miesto s chybou dostatočne malou či minimálnou
3. Efektívne sa tieto chyby dajú spočítať tak, že miesto minima zo sumy druhých mocnín rozdielov sa hľadá maximum zo sumy súčinov.
4. Sumy súčinov môžeme totiž počítať pomocou cirkulárnej konvolúcie, tj. vziať obraz a vzor, urobiť z nich Fourierovu transformáciu, vynásobiť ich spektrá a urobiť inverznú Fourierovu transformáciu

Det. obj. fázovou koreláciou

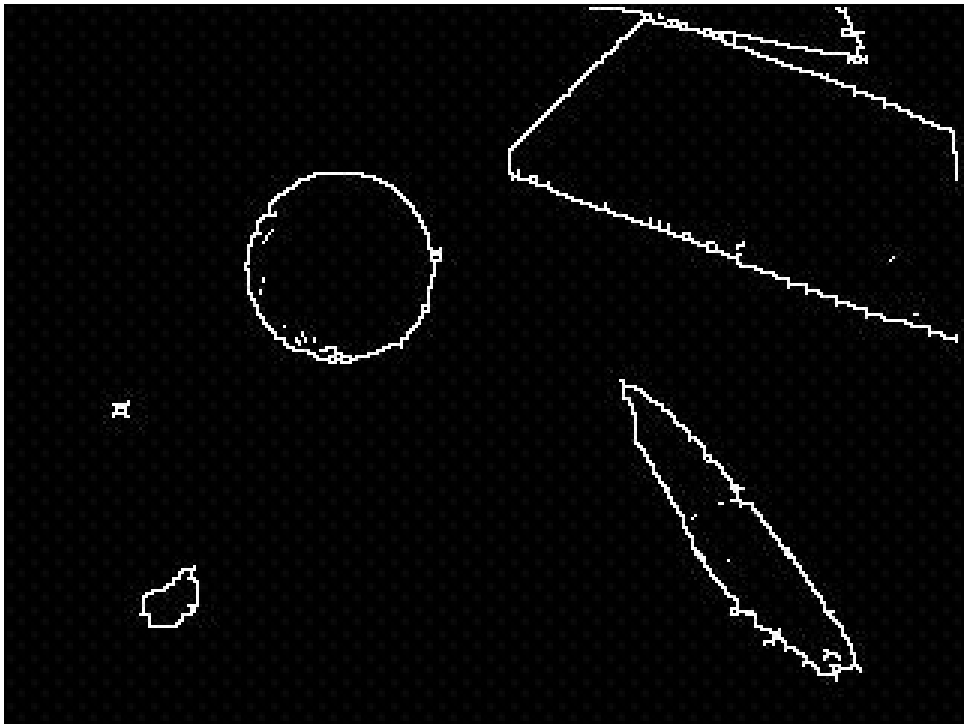
Det. obj. Feature Detector-om

1. Pomocou detektora význačných bodov dostaneme sadu bodov (a ich deskriptorov) zo vzoru a obrazu
2. (detektory SIFT, SURF, ORB tieto body hľadajú ako extrémy v trojrozmernom priestore, kde základňu tvorí obraz a poschodia nad ním čoraz viac rozmazaný obraz. Pritom neskúšajú len pôvodnú veľkosť ale obraz viacnásobne podvzorkujú)
3. Snažíme sa napárovať body vzoru na obraz (RANSAC), pokiaľ sa to dostatočne pekne podarí, zdetekovali sme objekt

Det. obj. Feature Detector-om



Detektory podľa tvaru



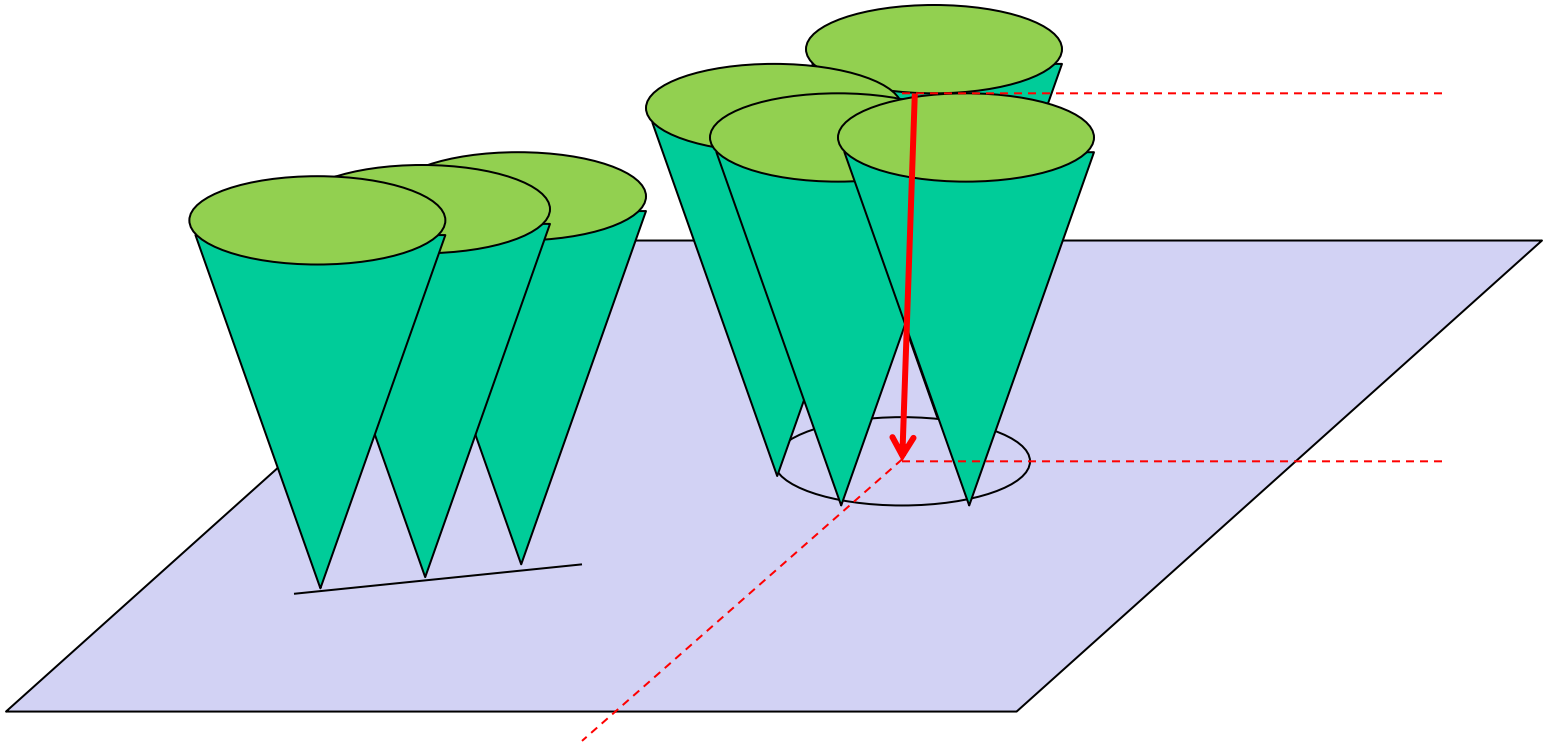
- Hrany poskytuje Canny operator

Detektory podľa tvaru

- Ad-hoc methods
- RANSAC
- *Hough transform*

Houghova transformácia

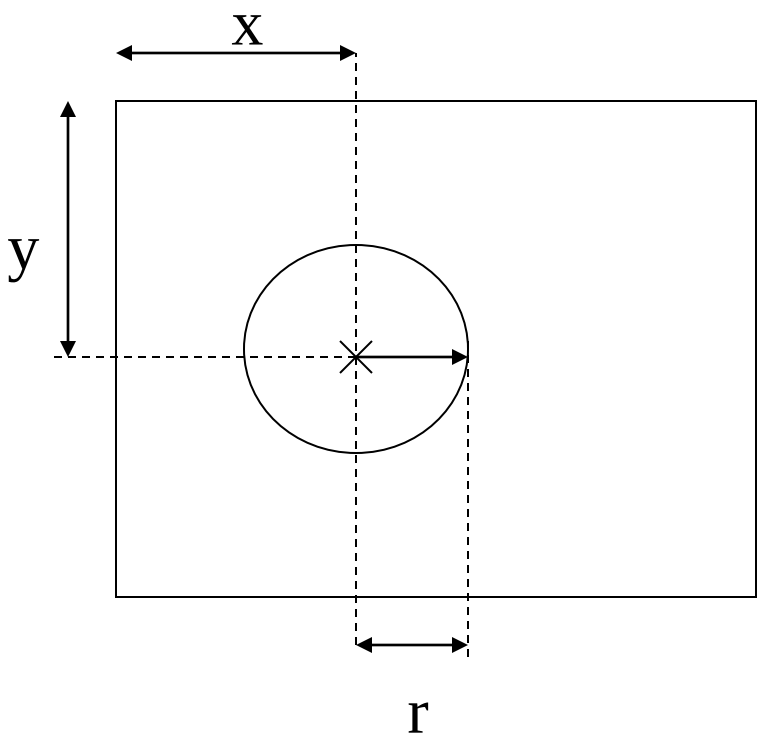
Každý pixel hrany hlasuje za všetky možné napr. kružnice, ktorých by mohol byť súčasťou. Tieto hlasy sa spočítajú a tá sada parametrov kružnice, ktorá získa dostatočne veľa hlasov, je zobrazená ako kružnica



Houghova transformácia

Vieme, že kružnicu môžeme reprezentovať tromi parametrami:

- x-súradnica stredu
- y-súradnica stredu
- polomer



Houghova transformácia

Každý z týchto parametrov má určitý rozsah

Napríklad keď máme obraz 320 x 240 bodov

- x-súradnica stredu má rozsah 0..319
- y-súradnica stredu má rozsah 0..239
- polomer má rozsah povedzme 10..200

a stačí ich brať ako celočíselné (lebo nemožeme dosiahnuť presnosť väčšiu než má kamera)

Houghova transformácia

- Celú situáciu nám teda bude reprezentovať celočíselné pole P[0..319,0..239,10..200] kde P[x,y,r] znamená:

„P[x,y,r] svedkov tvrdí, že na obraze je kružnica so stredom [x,y] a polomerom r“

Kto budú títo svedkovia ?

Houghova transformácia

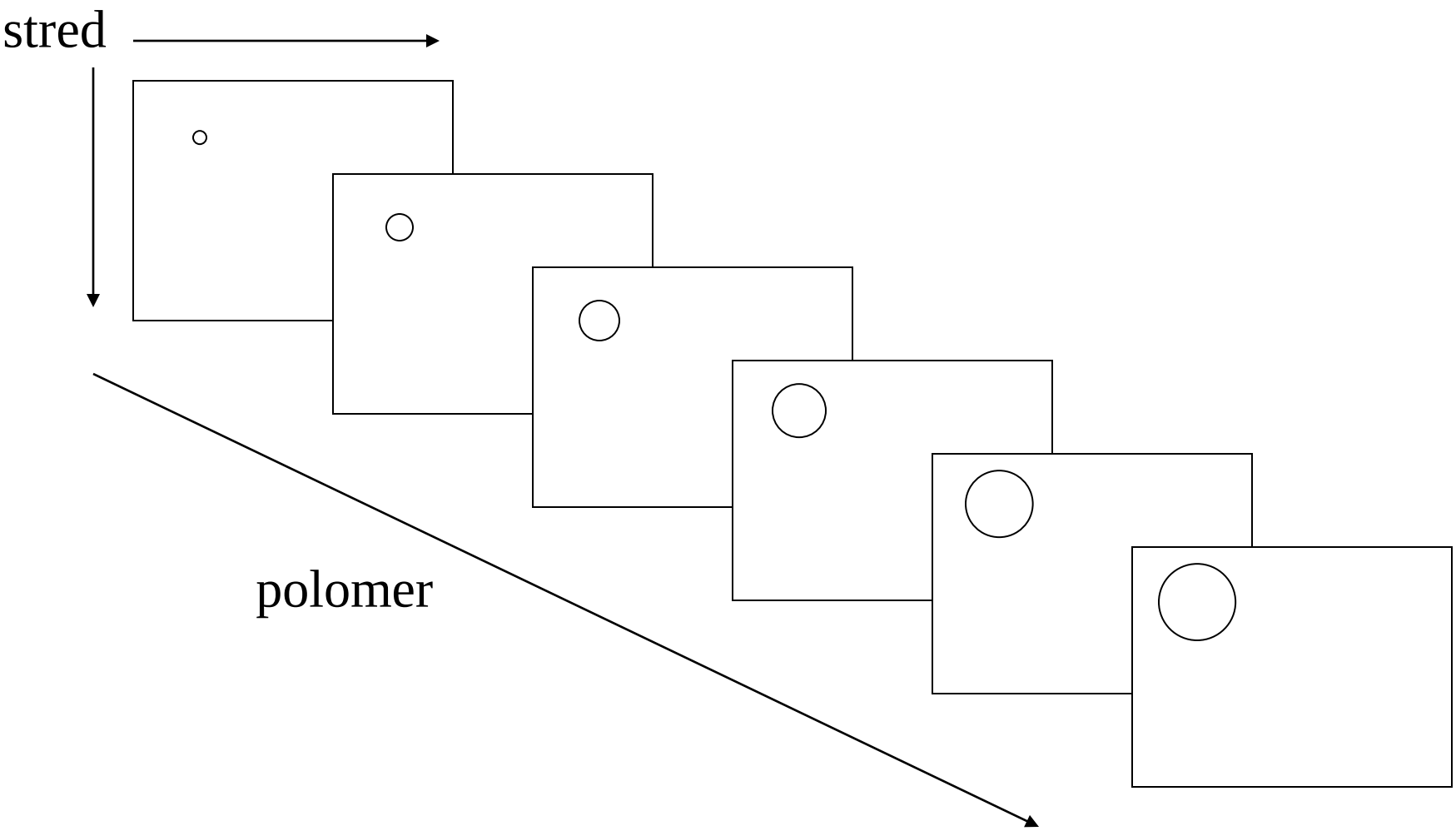
Nebude to nik iný ako body obrazu. Každý biely bod [x,y] na vyhranenom obraze si totiž namýšla že leží na hľadanej kružnici a svedčí:

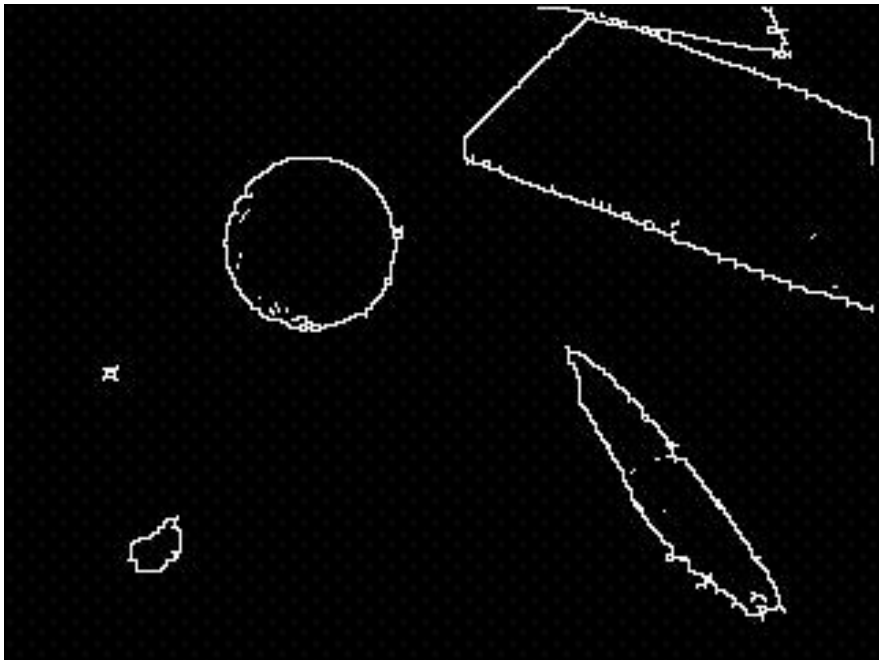
- Na obraze je kružnica so stredom [x,y] a polomerom 0
- Na obraze je kružnica so stredom [x-1,y] a polomerom 1
- Na obraze je kružnica so stredom [x+1,y] a polomerom 1
- Na obraze je kružnica so stredom [x,y-1] a polomerom 1
- Na obraze je kružnica so stredom [x,y+1] a polomerom 1
- Na obraze je kružnica so stredom [x-2,y] a polomerom 2
- …. atď

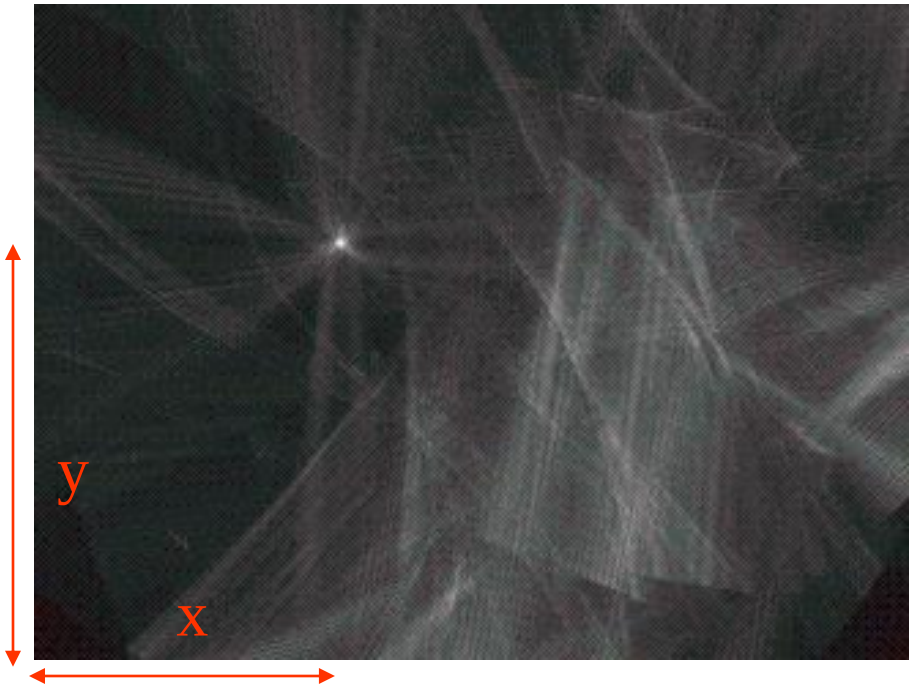
Houghova transformácia

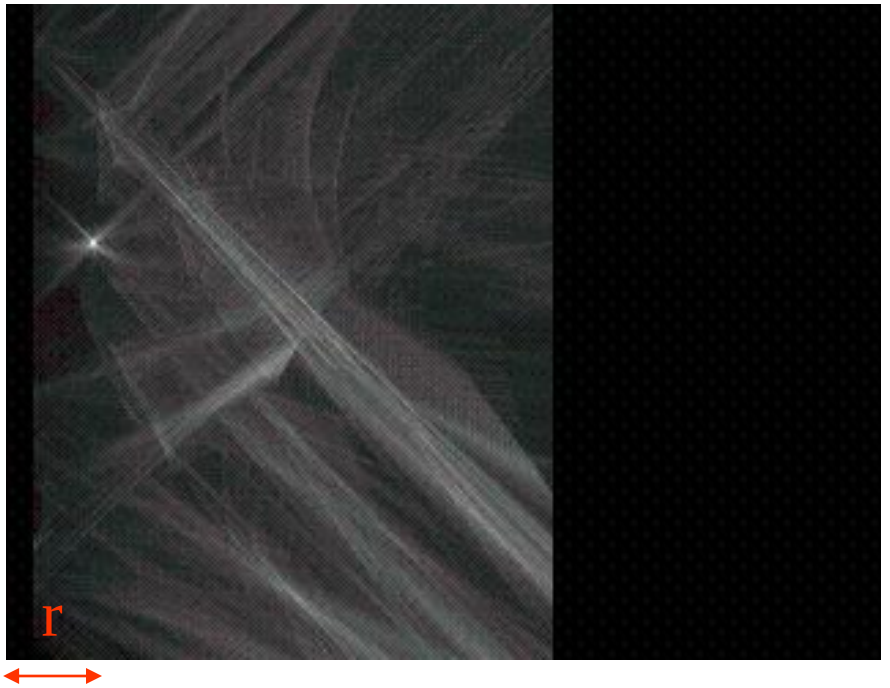
- vo všeobecnosti svedčí bod [x,y] pre všetky kružnice so stredom x+a, y+b a polomerom r, kde $a^2 + b^2 \cong r^2$
- Ako nájsť pravdu z toľkých tvrdení tak nepoľahlivých svedkov? Zistíme, ktorý názor sa najviac zhoduje!
- Pre každý biely bod na vyhranenom obraze teda prebehneme pole P a o jednotku zvýšime hodnotu všetkým za koho tento bod svedčí. Na záver sa pozrieme pre koho svedčí najviac bodov.

Houghova transformácia



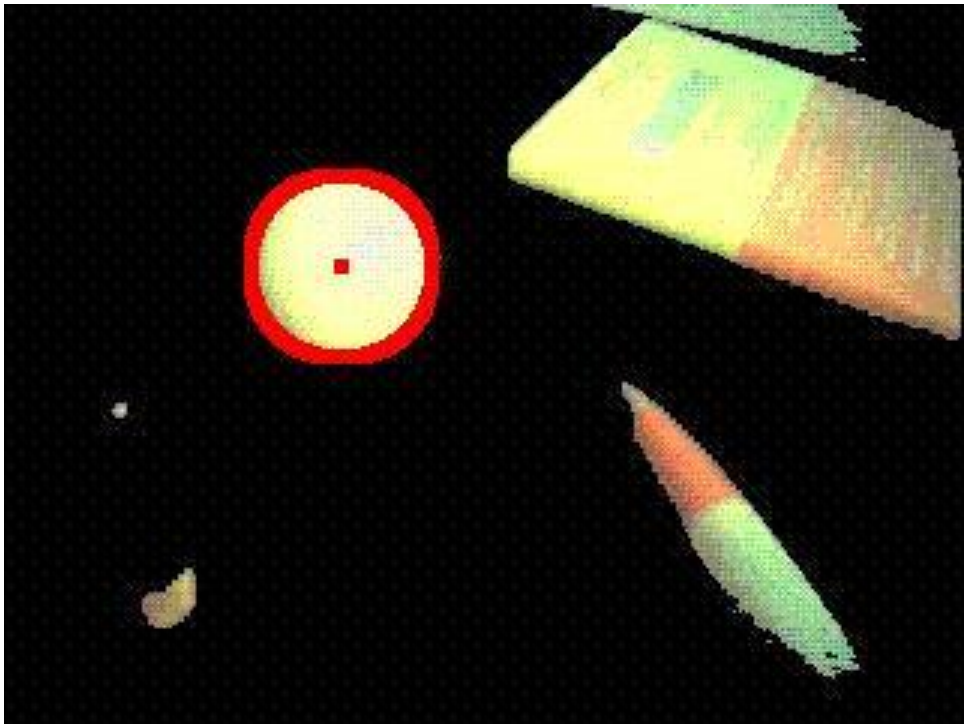






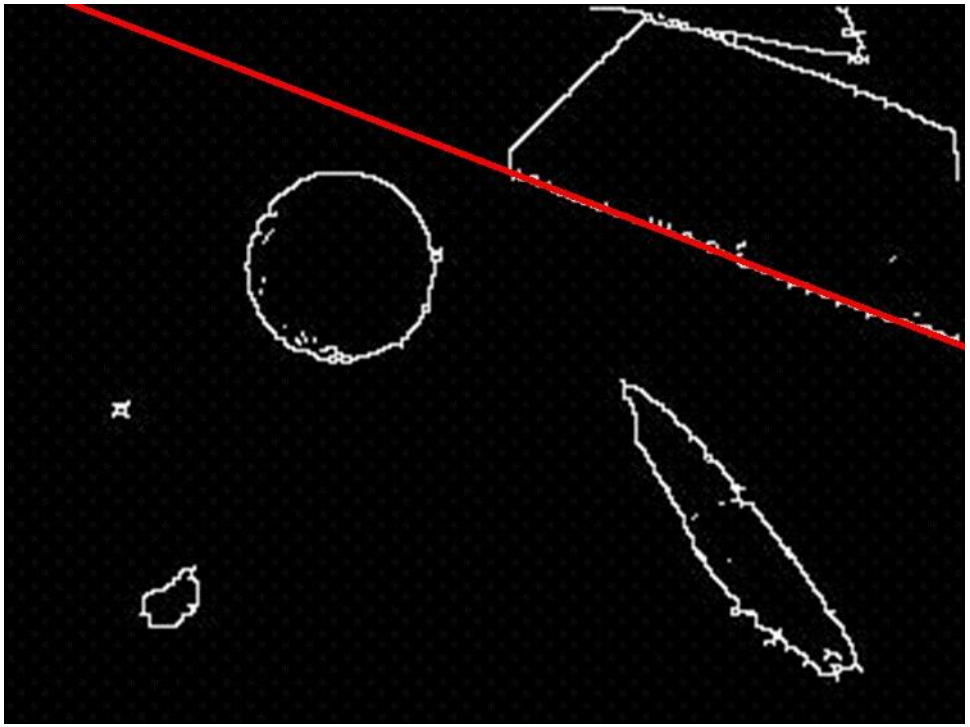
Takto transformujeme pôvodný obraz na niečo trojrozmerné.

Houghova transformácia



A maximum je hľadaný objekt

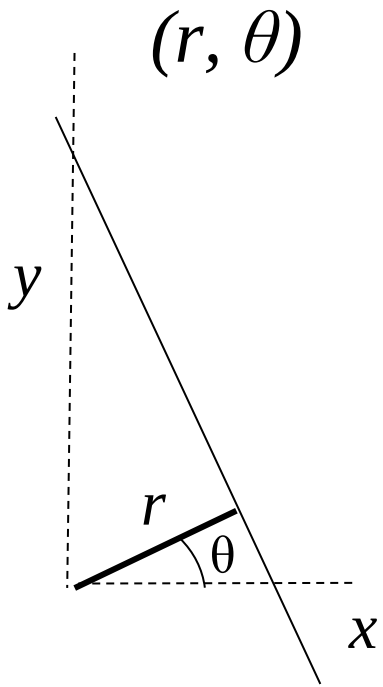
Houghova transformácia (priamky)



Podobne môžeme nájsť priamky (2 parametre),
elipsy (4 parametre) ,…

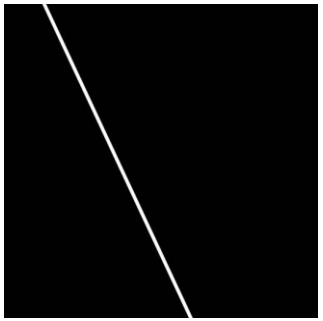
Hough transform

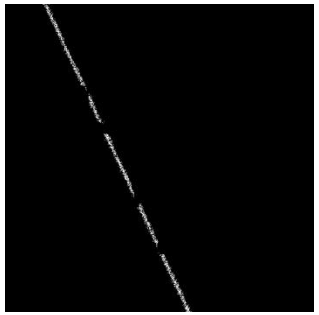
$$r = x \cos\theta + y \sin\theta$$



Rendering

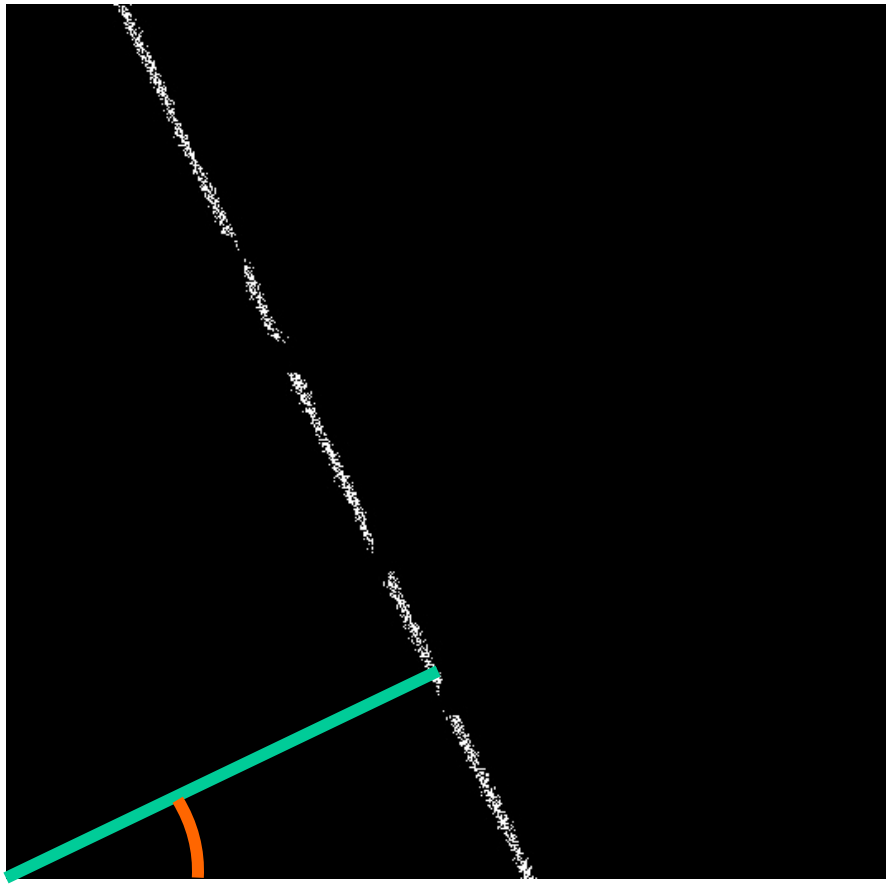
Hough transform



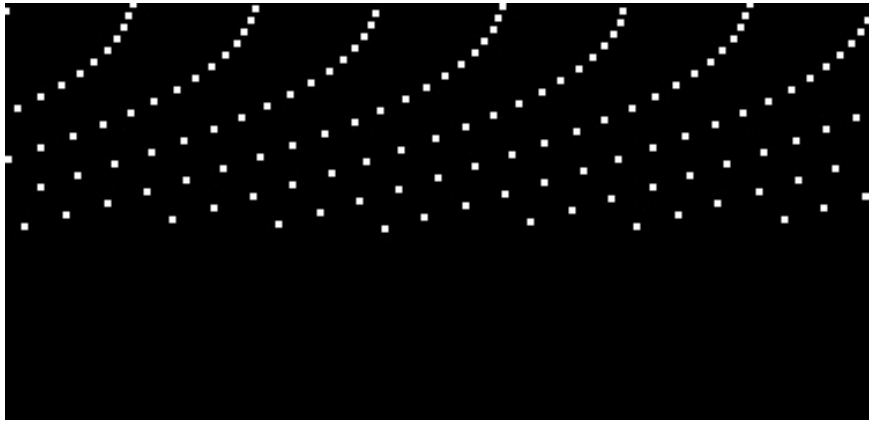


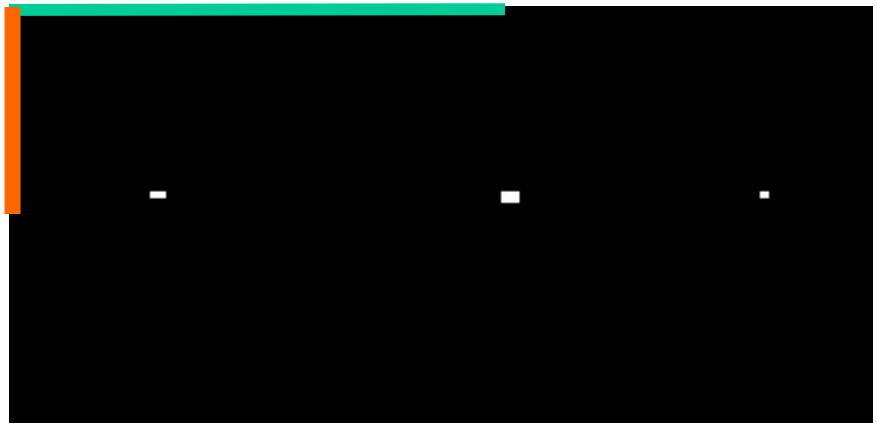
Hough transform

voters



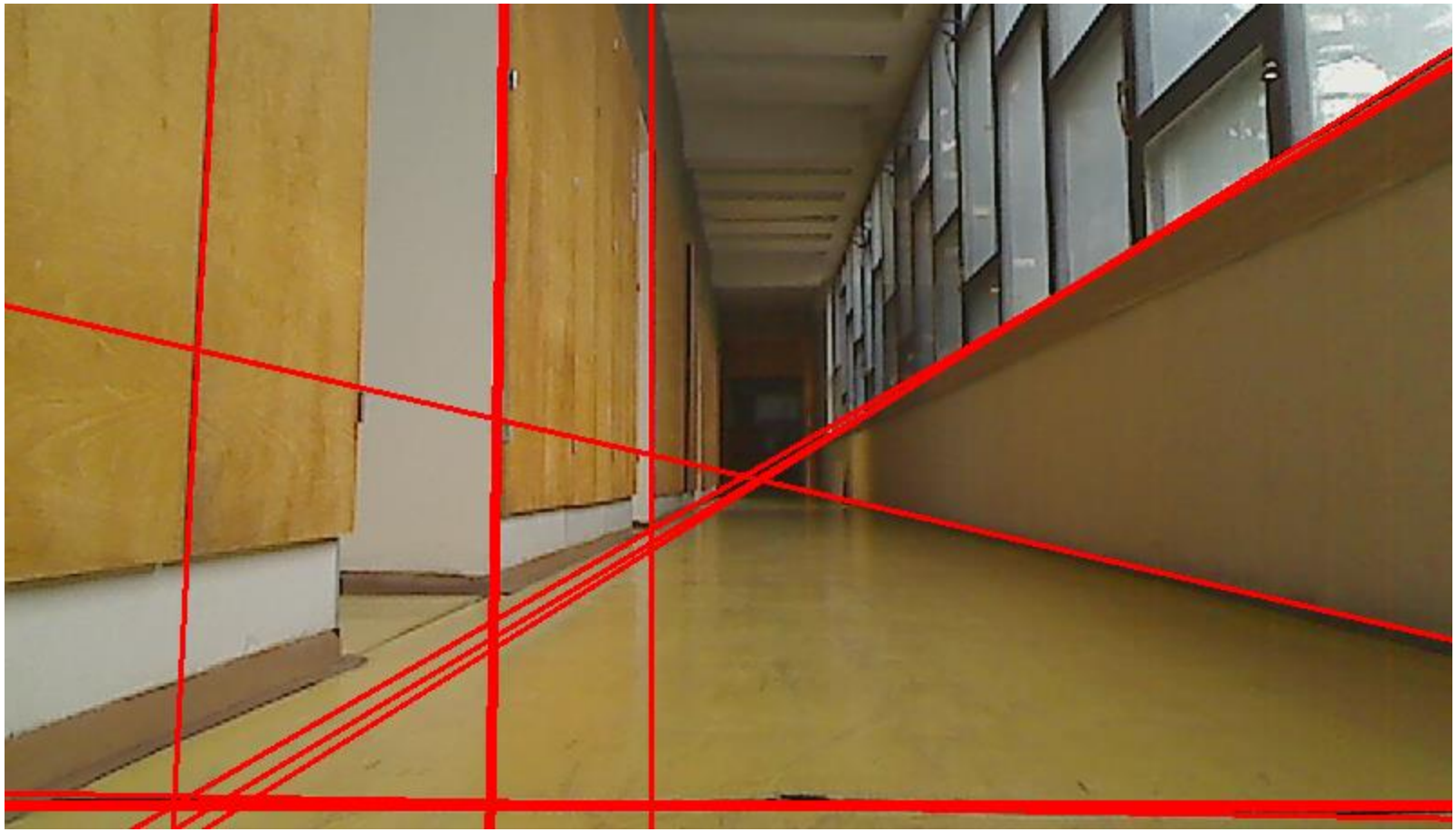
votes of one voter





votes summary

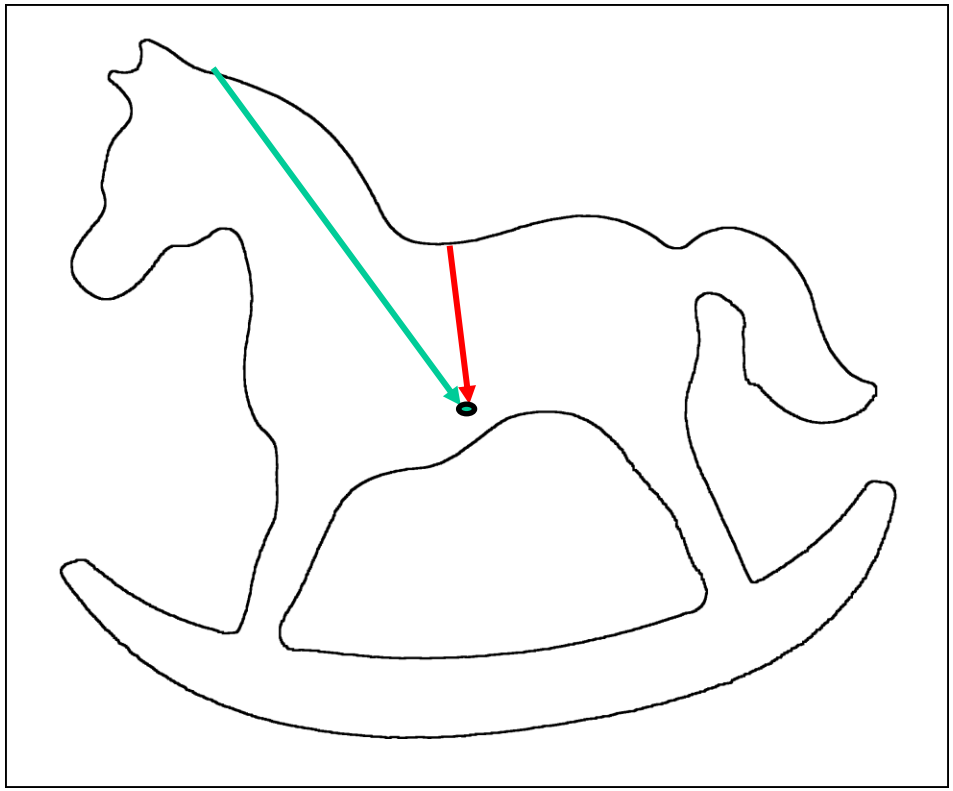
Hough transform - lines



Hough transform - segments

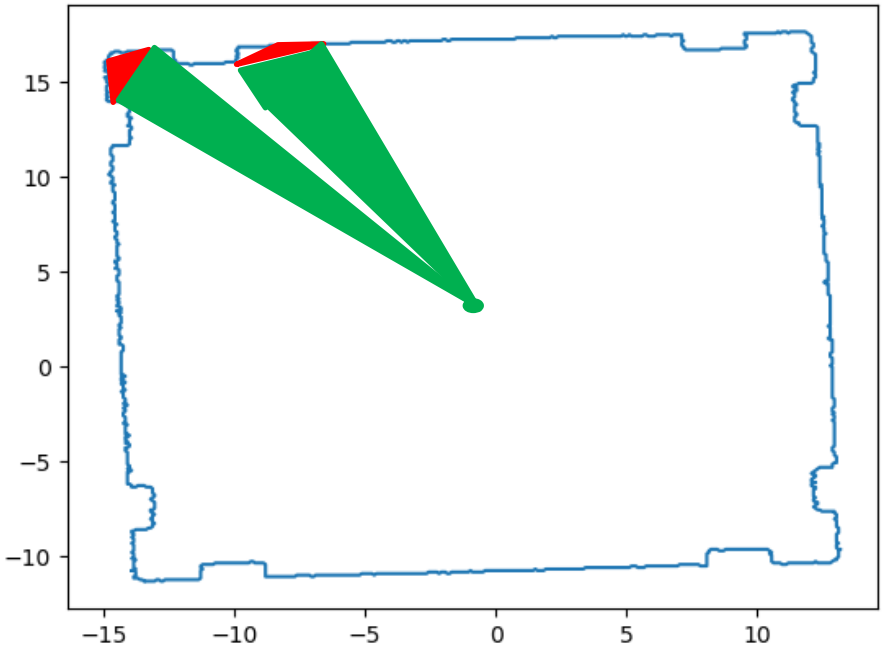
Vanishing point detection

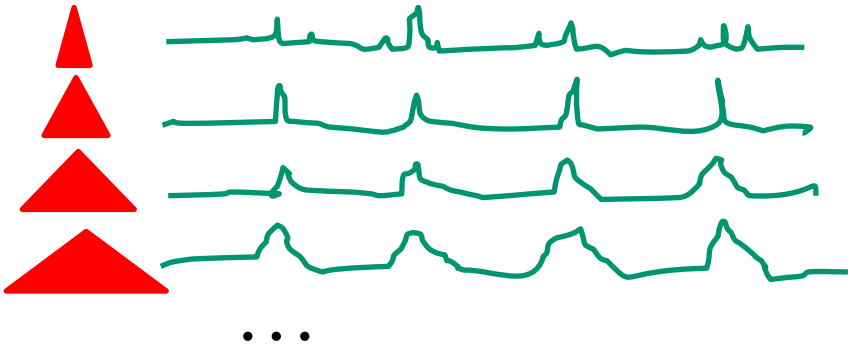
Generalizovaná Houghova transformácia



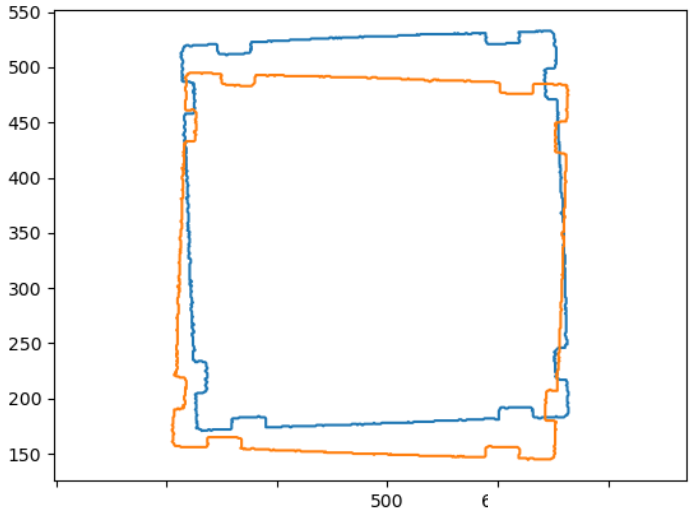
Analogicky môžeme nájsť objekt s definovaným tvarom (posunutý, zmenšený a zväčšený – rovnako v x a y)

Triangle Area Representation

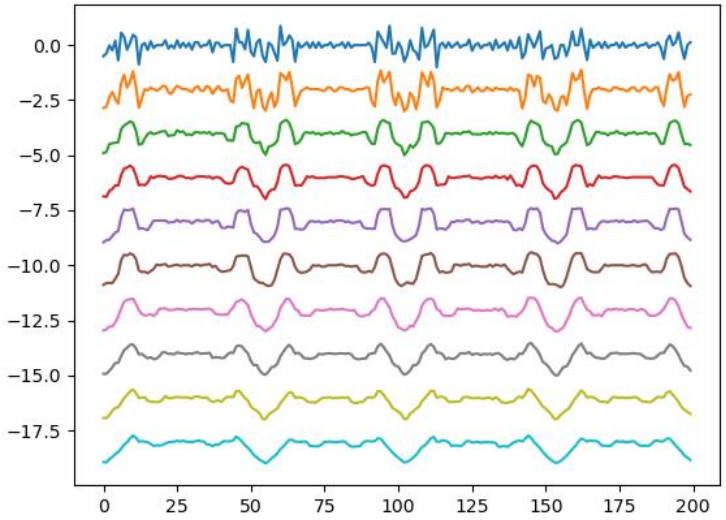


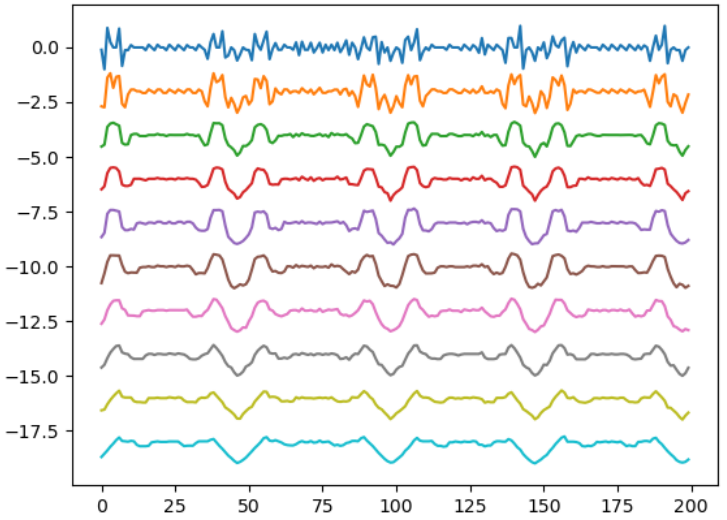


Reprezentácie dvoch objektov toho istého druhu budú podobné, ale posunuté

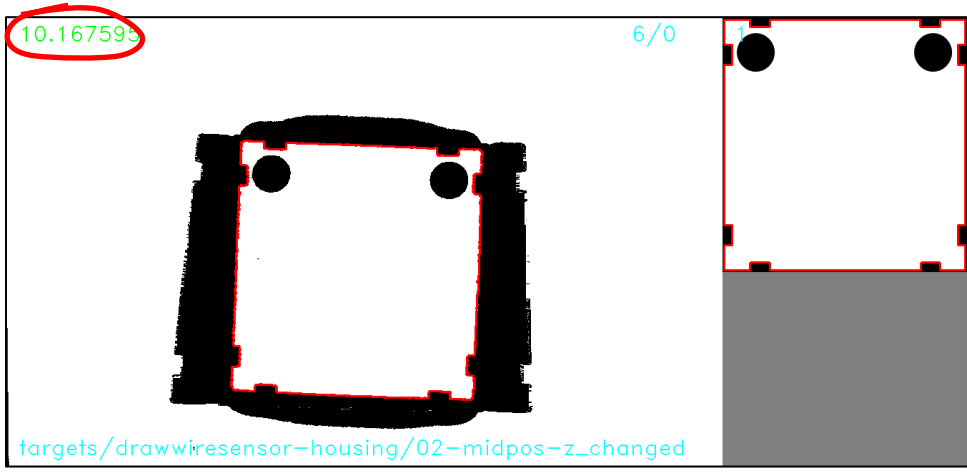


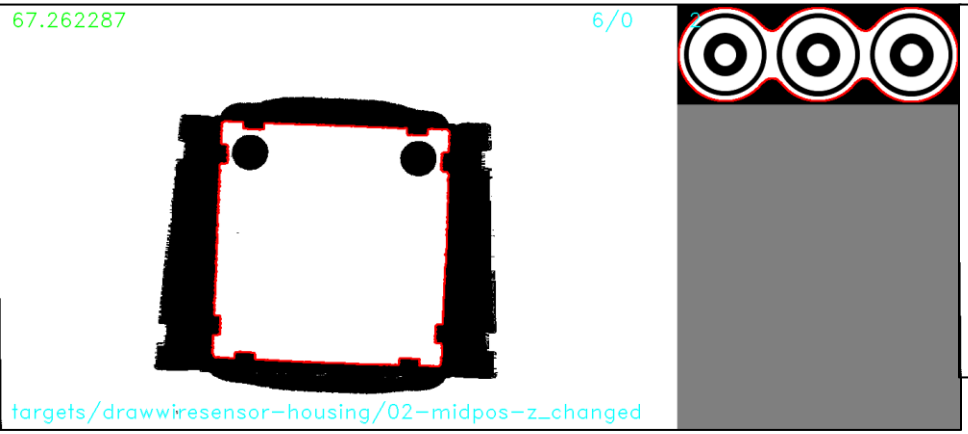
Toto posunutie zistí algoritmus cyklický DSW (dynamické programovanie)

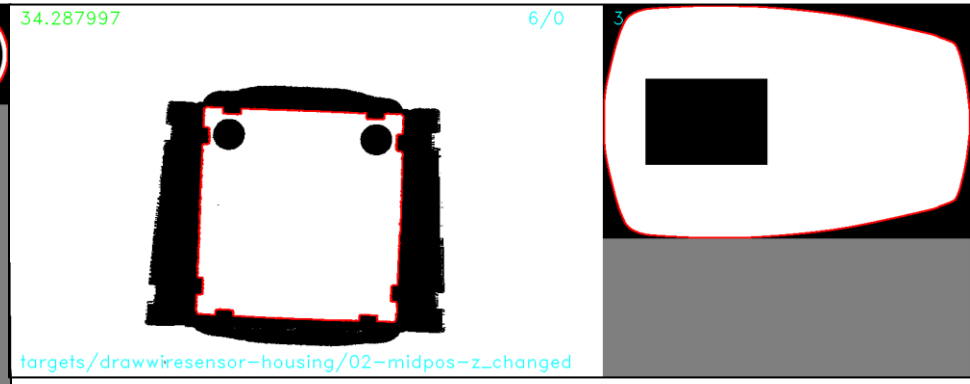




Výhoda TAR: vystačíme s jediným vzorom

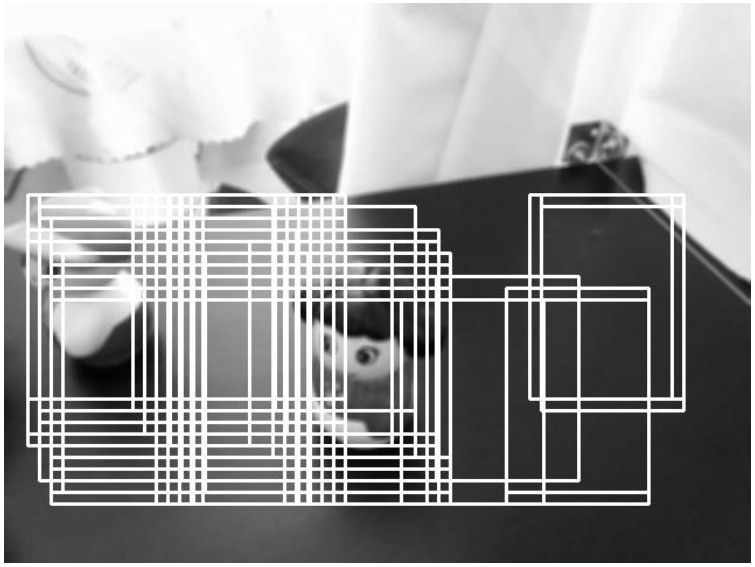






klasifikátor → detektor

- Jadrom tradičného detektora je klasifikátor, ktorý je spustený na každé možné miesto výskytu objektu v rôznych veľkostiach:
- **Sliding window algorithm**

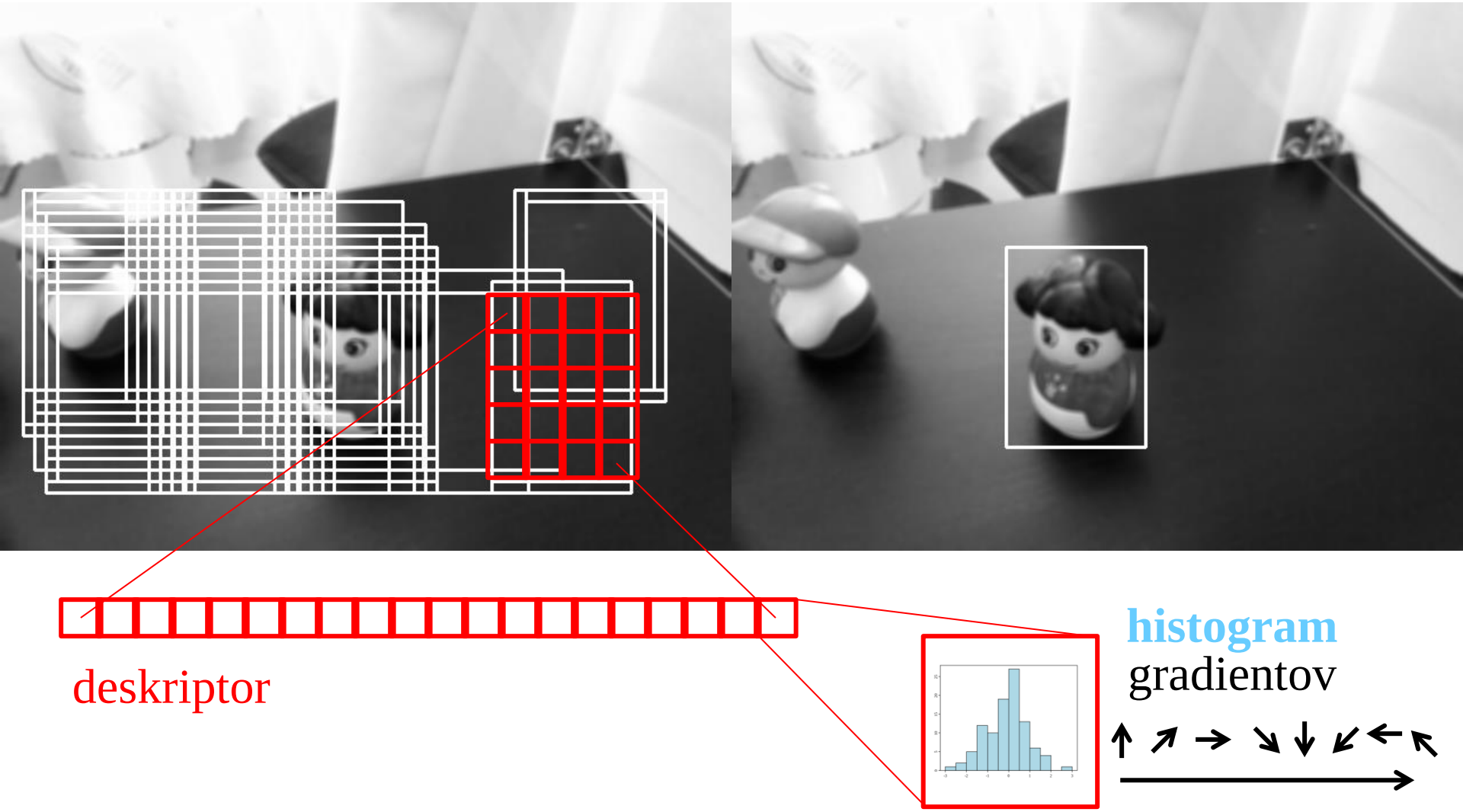


HOG detektor

1. Vybraný objekt sa pokryje sieťou regiónov
2. V každom regióne sa spočíta histogram gradientov.
3. Súbor týchto histogramov zo všetkých regiónov tvorí deskriptor
4. Na danom obraze potom hľadáme oblasti s podobným deskriptorom
5. Spravidla potrebujeme sadu pozitívnych a negatívnych príkladov a učíme z nich klasifikátor, napríklad Support Vector Machine

HOG detektor

Používa Sliding window algorithm, HOG a SVM



LBPH

Hľadáme:
invariantné vyjadrenie obrazu

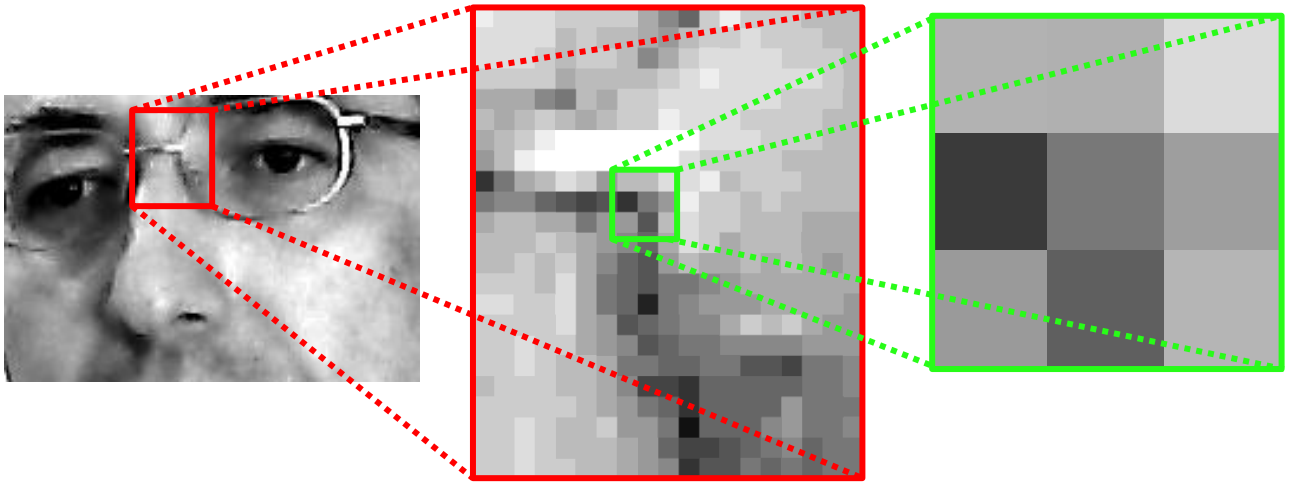





- Aká reprezentácia je rovnaká pre tieto obrázky?

Local Binary Patterns

- Invariant: ak je jeden pixel svetlejší od druhého na jednom obraze, bude tento vzťah zachovaný aj obraze so zmenenou jasnosťou, kontrastom či ostrosťou.



0	0	0
1		0
0	1	0

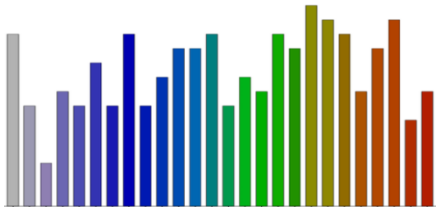
`00000101`

5

- Každému pixelu vieme priradiť jeho LBP – číslo od `0` do `255`

Local Binary Pattern Histogram

- Každý pixel má svoj LBP – číslo od `0` do `255`



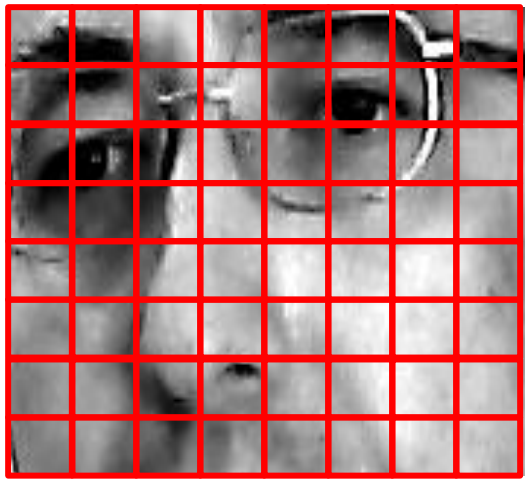
- Spočítame koľko je akých: napr. 587 krát 0, 32 krát 1, 564 krát 2, ani raz 3, ... a dva krát 127, dostaneme histogram

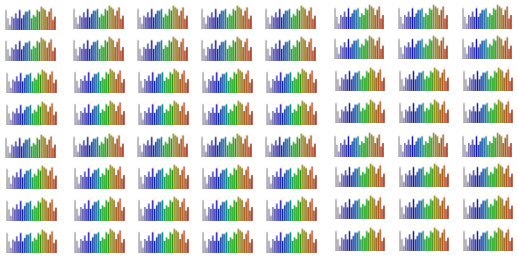
`3F 2E 01 ... FF`

256 bytov

- Tento histogram normalizujeme tak, že jedným bytom vyjadríme relatívnu početnosť

Local Binary Pattern Histogram





3F 2E 01 03 ... FF

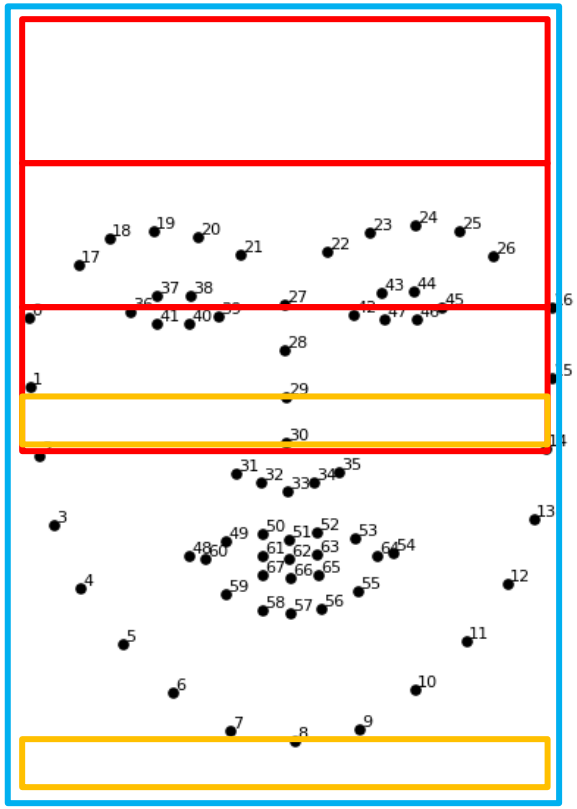
16384 bytov

- Aby sme zachytili aj priestorové rozloženie, môžeme obraz štruktúrovať na oblasti napr. obraz 64x64 pokryjeme mriežkou 8x8 a pre každú časť spočítame jej LBP, tieto potom spojíme do vektora za sebou, čiže v tomto prípade dostaneme 64 x 256 = 16384 bytov

Local Binary Pattern Histogram

- LBPH sú len málo odolné voči posunutiu a vôbec nie voči otočeniu či zväčšeniu/zmenšeniu
- Obraz preto musíme vzhľadom na tieto transformácie normalizovať vo vlastnej réžii (napríklad pri tvárach pomocu detektora tvárových čŕt)
- Potom stačí vyrovnaný a otočený výrez zmenšiť na štandardnú veľkosť (napr. 64x64), čím riešime zväčšenie a zmenšenie

Predspracovanie



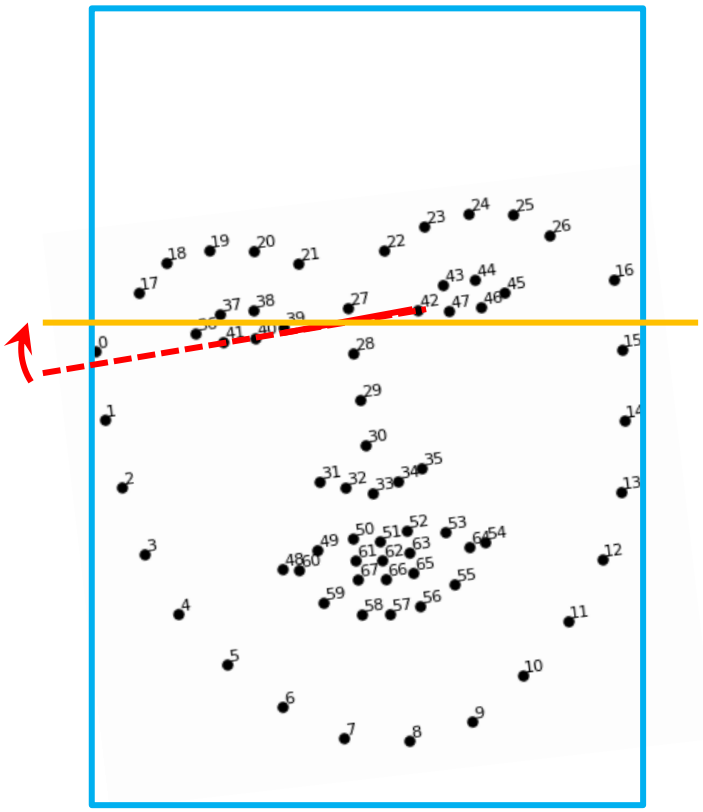
FaceAligner

HOG

HOG

HOG

Predspracovanie



- normalizujeme veľkosť na 64 x 64
- normalizujeme rotáciu
- posunutie nás netrápi

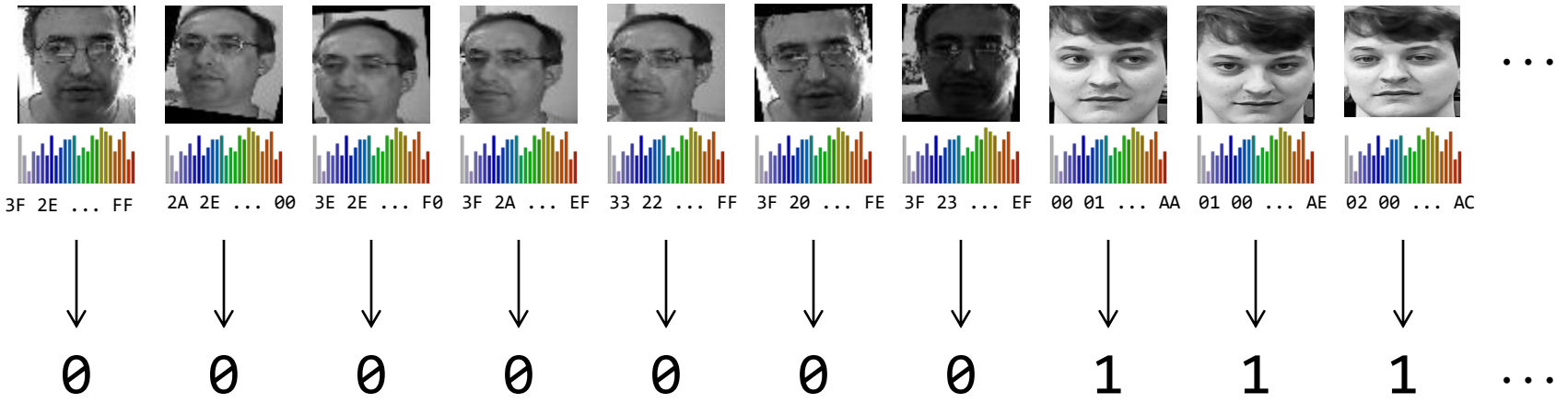
F.L.

F.L.

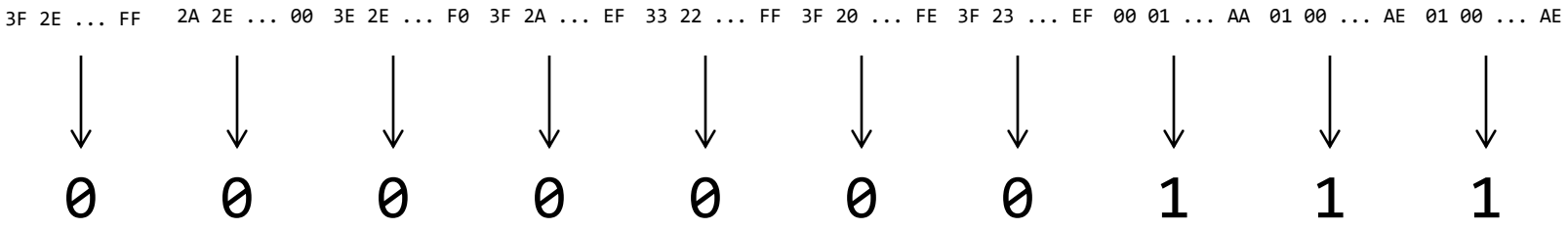
F.L.

Dataset pre Machine learning

- z určitého pohľadu na konkrétnu tvár teraz vieme dostať približne rovnaký LBPH bez ohľadu na to, či je bližšie alebo ďalej od kamery, či je naklonená vľavo, vpravo alebo je rovno
- Iný LBPH dostaneme, keď zmeníme uhol pohľadu na tvár. Jednu tvár nám teda bude reprezentovať viacero snímok tváre z rôznej strany, a toľko isto LBPH-ov



Trénovanie



Sady fotiek roztriedené podľa osoby na fotke spracujeme na skupiny LBPH, každej skupine priradíme určité ID a pomocou strojového učenia natrénujeme model klasifikátora, ktorý tieto osoby rozlišuje.

Klasifikátor datasetu LBPH môže byť založený na:

- LDA (fisherfaces)
- SVM

Rozpoznávanie

Tento model môžeme potom použiť tak, že v každom obrázku z kamery hľadáme tvár, k nej príslušný LBPH a ten vložíme do modelu nech povie jeho ID.