Tema 8. Detección de líneas y esquinas

4730 Visión Industrial
Ingeniería Técnica Industrial
especialidad en Electrónica Industrial

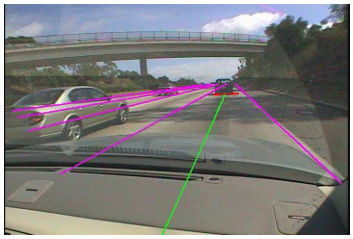
Universitat de les
Illes Balears
Departament de Ciències
Matemàtiques i Informàtica

Alberto ORTIZ RODRÍGUEZ

Indice

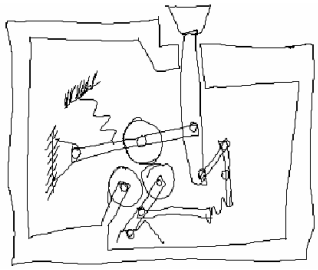
Introducción

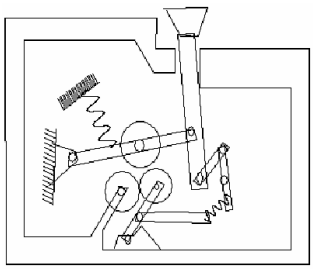
- Detección de objetos rectilíneos / segmentos rectos:
  - Detección de las líneas de una carretera en una aplicación de navegación visual autónoma



Introducción

- Detección de objetos rectilíneos / segmentos rectos:
  - Reconstrucción de un diseño hecho a mano

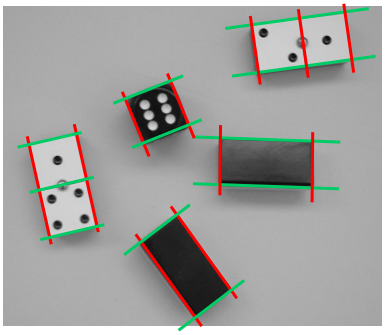


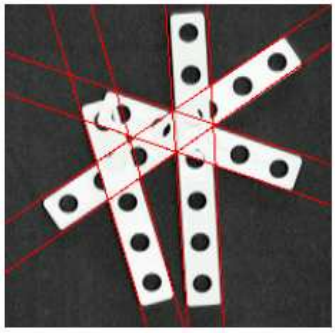


  - Otras aplicaciones …

Introducción

- Aparte, líneas y esquinas son elementos importantes de muchos objetos, sobre todo en objetos artificiales y principalmente de su contorno ⇒ reconocimiento de objetos



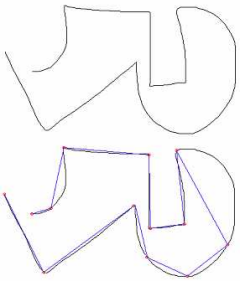


Introducción

- Aplicaciones en robótica:

Introducción

- Además, los contornos de los objetos naturales pueden ser poligonalizados
  ⇒ esquinas y líneas



Introducción

- En visión, se utiliza la categoría ***image features*** (características de la imagen) para hacer referencia a píxeles o grupos de píxeles que verifican cierta propiedad y que suelen servir de referencia para realizar tareas visuales de más alto nivel [Introductory techniques for 3D computer vision – Trucco, Verri]
  ⇒ *features* = {bordes, **esquinas**, **líneas**, circunferencias, elipses, texturas, etc…}

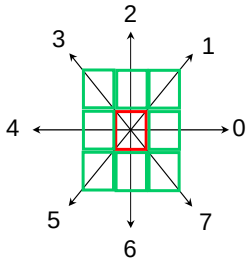
- Partimos de los **bordes** de la imagen

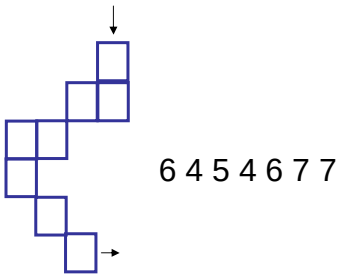
Indice

- Introducción
- Detección de segmentos rectos mediante códigos de cadena
- Transformada de Hough
- Detección de esquinas

Detección de segmentos rectos mediante códigos de cadena

- Dado un conjunto de píxeles de borde adyacentes, se analiza cada pareja de 2 píxeles adyacentes y se codifica dicha pareja de acuerdo con su orientación:

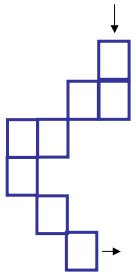




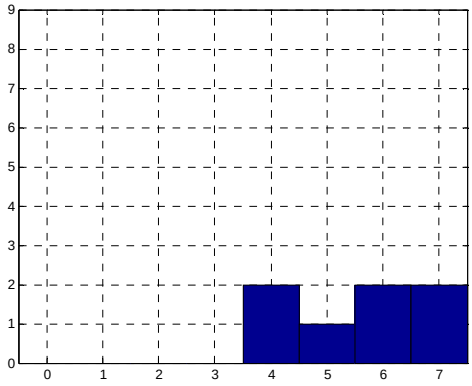
6 4 5 4 6 7 7

Detección de segmentos rectos mediante códigos de cadena

- Una vez obtenida la codificación, se crea un histograma de orientaciones:

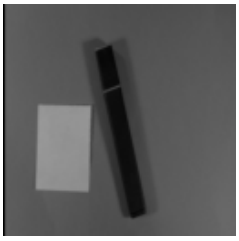


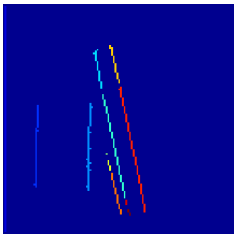
6 4 5 4 6 7 7

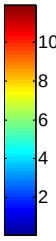


Detección de segmentos rectos mediante códigos de cadena

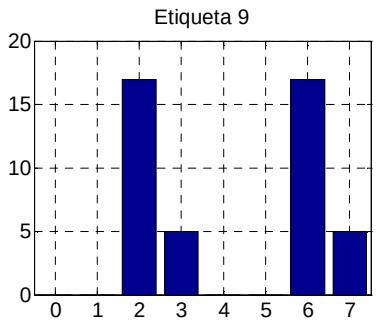
- Otro ejemplo:








```
img = imread('Fig_0.gif');
bordes = edge(img,'sobel',[],'vertical');
L = bwlabel(bordes);
figure; imshow(img);
figure; imshow(bordes);
figure; imagesc(L); colorbar;
```



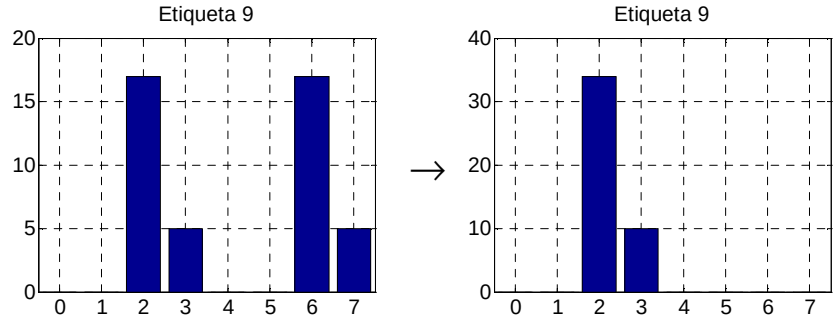
Detección de segmentos rectos mediante códigos de cadena

- Código MATLAB:

```
of = [+0 +1; -1 +1; -1 +0; -1 -1; +0 -1; +1 -1; +1 +0; +1 +1];
[v,h] = size(img);
H = zeros(1,8);
for j = 1:v
    for i = 1:h
        if L(j,i) == 9
           for k = 1:8
               jj = j + of(k,1); ii = i + of(k,2);
               if L(jj,ii) == 9, H(k) = H(k)+1; end
           end
           L(j,i) == 0;
         end
    end
end
figure; bar(0:7,H); axis([-0.5 7.5 0 20]); grid on;
set(gca,'FontSize',20); title('Etiqueta 9');
```


Detección de segmentos rectos mediante códigos de cadena

- Las direcciones 0, 1, 2 y 3 son las mismas que, respectivamente, las direcciones 4, 5, 6 y 7




```
H2 = H; H2(1:4) = H2(1:4) + H2(5:8); H2(5:8) = [0 0 0 0];
```

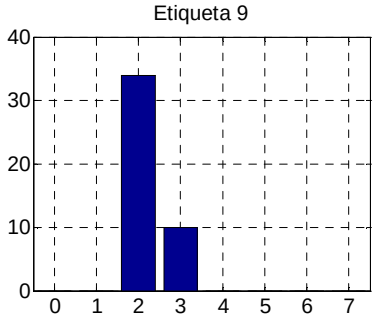
Detección de segmentos rectos mediante códigos de cadena

- A partir del histograma, según el número de barras N:
  - SI N = 1 ENTONCES la línea es recta
  - SI N = 2 ENTONCES
    SI las barras son adyacentes ENTONCES
      SI la barra menor es inferior a T ENTONCES la línea es recta
      SINO la línea no es recta (*)
    SINO la línea no es recta (*)
  - SI N = 3 ENTONCES
    SI las barras son adyacentes Y la barra central es la más larga Y las otras barras son inferiores a T ENTONCES la línea es recta
    SINO la línea no es recta (*)
  - SI N = 4 ENTONCES la línea no es recta (*)

(*) hay dos orientaciones que difieren al menos en 90º

Detección de segmentos rectos mediante códigos de cadena

- Ejemplo anterior:



(1) N = 2

(2) barras adyacentes

(3) T ≥ 10 → la línea es recta
T < 10 → la línea no es recta

Indice

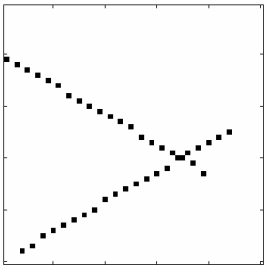
- Introducción
- Detección de segmentos rectos mediante códigos de cadena
- Transformada de Hough
- Detección de esquinas

Transformada de Hough

- La Transformada de Hough (HT) fue propuesta por Paul Hough en 1962 y patentada por IBM.
- Ha resultado ser una herramienta estándar en visión por computador para la detección de líneas rectas, círculos y elipses.
- La HT es particularmente robusta a datos incompletos y contaminados.
- Sirve para detectar cualquier número de líneas (u otros objetos geométricos que admitan descripciones paramétricas, p.e. $\{(x,y) \mid y = ax+b, a, b \in R\}$ [recta] o $\{(x,y) \mid (x-x_0)^2 + (y-y_0)^2 = r^2, x_0,y_0,r \in R\}$ [circunferencia]).

Transformada de Hough

- Consideramos la detección de líneas.
- La imagen siguiente es una representación idealizada de un conjunto de píxeles identificados como bordes.



- Se trata de identificar de forma automática los segmentos rectos contenidos en la imagen.

Transformada de Hough

- Una línea es un objeto geométrico que puede ser descrito por la siguiente ecuación:

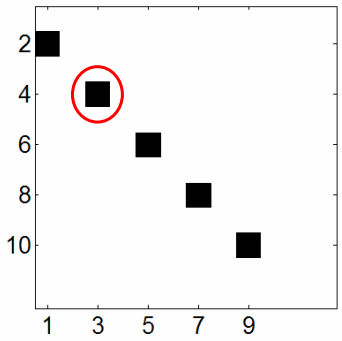
$$\{(x,y)|y = ax + b, a, b \in \mathbb{R}\}\ (slope\text{-}intercept\ form)$$

- Se trata de encontrar parejas (a,b) correspondientes a las líneas que contienen los segmentos rectos de la imagen.
- La HT parte de la siguiente variación de la ecuación anterior:

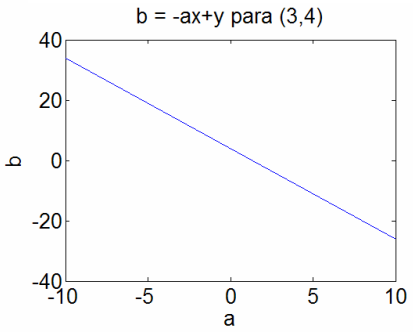
$$y = ax + b \longrightarrow b = -ax + y$$

Transformada de Hough

- Ahora, para un cierto píxel de borde (x_i,y_i), consideramos todos los valores que puede tomar el parámetro ‘a’ y calculamos el valor del correspondiente parámetro ‘b’:



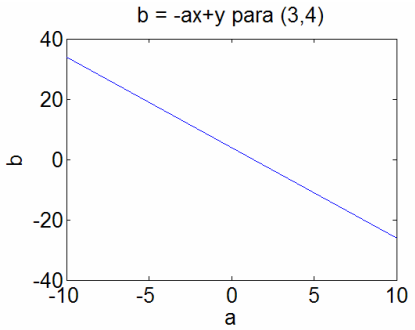
p.e. (x,y) = (3,4): a = -10:0.01:10,
b = -a*x + y = -a*3 + 4



Transformada de Hough

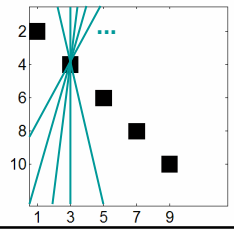
- Ahora, para un cierto píxel de borde (x_i,y_i), consideramos todos los valores que puede tomar el parámetro ‘a’ y calculamos el valor del correspondiente parámetro ‘b’:

(x,y) = (3,4): a = -10:0.01:10,
b = -a*x + y = -a*3 + 4



❖ Un píxel de borde genera, en el espacio de parámetros, una línea recta formada por las parejas (a,b) encontradas.

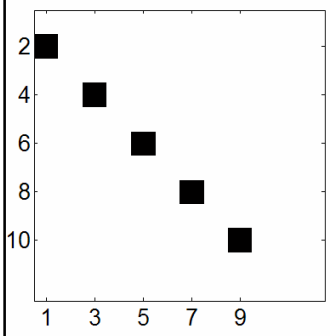
❖ Cada una de las parejas (a,b) corresponde a una de las rectas a las que podría pertenecer el punto (x,y) considerado.

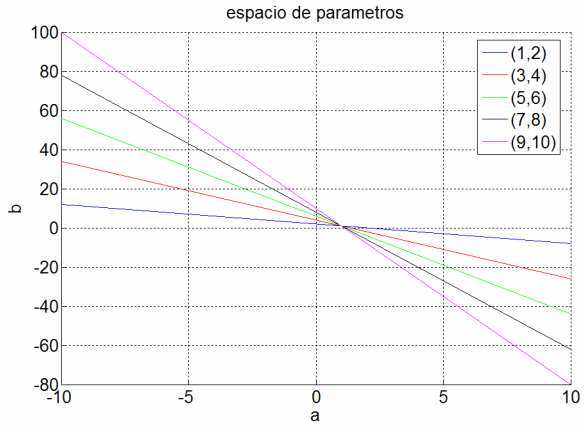


Transformada de Hough

- ¿Qué ocurre si consideramos todos los píxels de borde?

$\forall(x_i,y_i)$: a = -10:0.01:10, b = $-a*x_i + y_i$

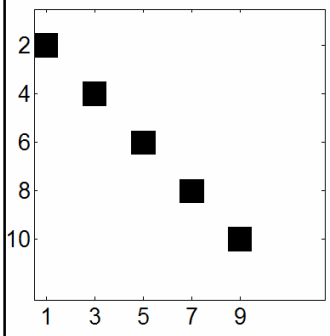


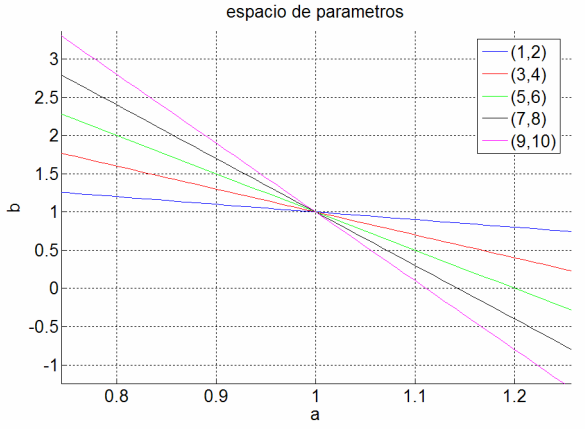


Transformada de Hough

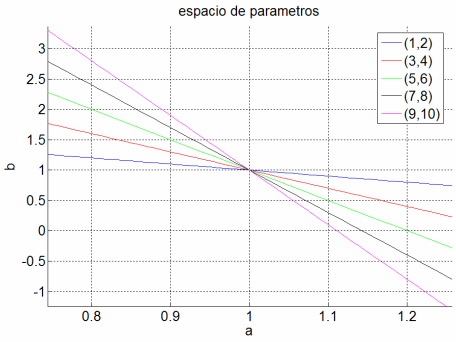
- ¿Qué ocurre si consideramos todos los píxels de borde?

$\forall(x_i,y_i)$: a = -10:0.01:10, b = $-a*x_i + y_i$ **(ZOOM sobre la intersección)**





Transformada de Hough



- Cada punto (x_i,y_i) ha generado una recta que contiene los parámetros (a,b) de todas las rectas a las cuales podría pertenecer.
- La intersección (a,b) = (1,1) corresponde a los parámetros de la recta a la que pertenecen todos los puntos de borde !!

• Por tanto, una forma automática de detectar líneas en una imagen es hacer que cada píxel de borde "vote" por las líneas a las que pertenecería. La pareja de parámetros (a,b) más votada sería la línea recta a seleccionar.

• En el ejemplo anterior, la pareja (a,b) = (1,1) habría recibido 5 votos y el resto sólo 1 voto.

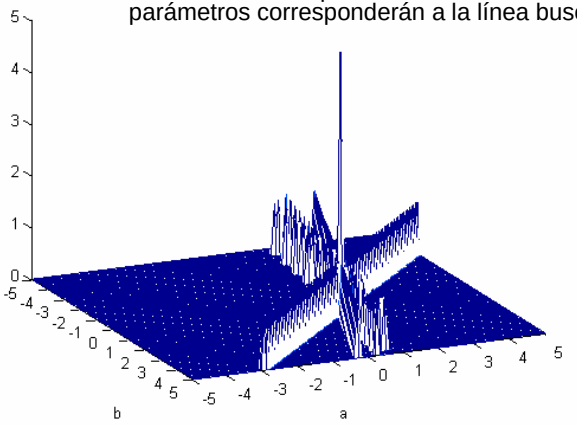
Transformada de Hough

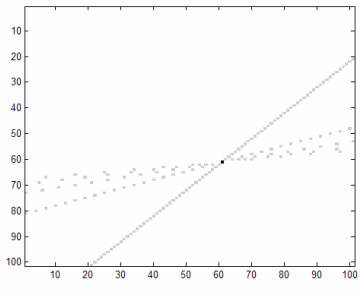
- Algoritmo:

1. Discretizar el espacio de parámetros en un conjunto de celdas (a_k,b_k) | $a_k \in a_0{:}\Delta a{:}a_n$, $b_k \in b_0{:}\Delta b{:}b_m$
2. Definir una matriz H_{sxt} con una celda por cada combinación posible a_k (filas) y b_k (columnas) [$s = (a_n-a_0)/\Delta a + 1$, $t = (b_m-b_0)/\Delta b + 1$]
3. $H(: , :) \leftarrow 0$
4. <u>para cada</u> píxel de borde (x_i,y_i)
   41. <u>para</u> $a_k = a_0{:}\Delta a{:}a_n$
      411. $b = -a_k x_i + y_i$
      412. Determinar el b_k más próximo a b
      413. $u = (a_k-a_0)/\Delta a + 1$, $v = (b_k-b_0)/\Delta b + 1$
      414. $H(u,v) = H(u,v) + 1$
   42. <u>fpara</u>
5. <u>fpara</u>

Transformada de Hough

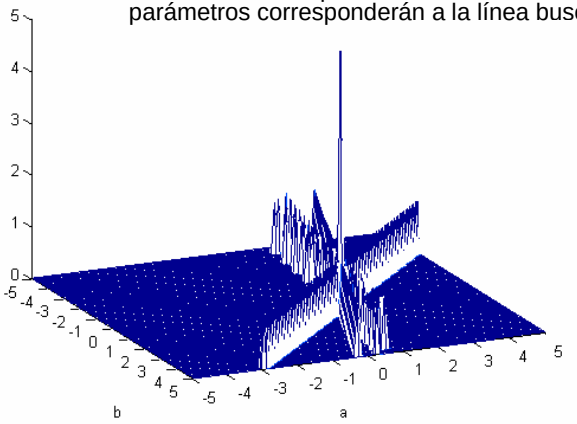
- Al final de la ejecución de este algoritmo, ¿qué habrá en la matriz H?
  - Habrá una celda que habrá acumulado más votos que las otras, cuyos parámetros corresponderán a la línea buscada

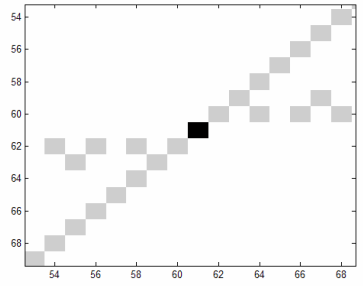




Transformada de Hough

- Al final de la ejecución de este algoritmo, ¿qué habrá en la matriz H?
  - Habrá una celda que habrá acumulado más votos que las otras, cuyos parámetros corresponderán a la línea buscada



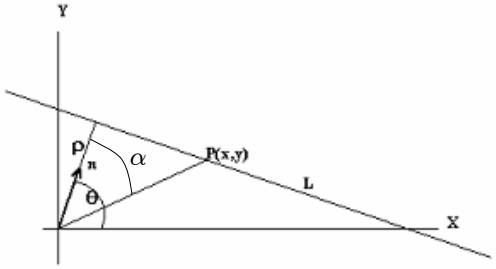


(ZOOM sobre la intersección)

Transformada de Hough

- PROBLEMA: a ∈ (- ∞, +∞)
- SOLUCIÓN: utilizar la **forma normal** de una recta

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

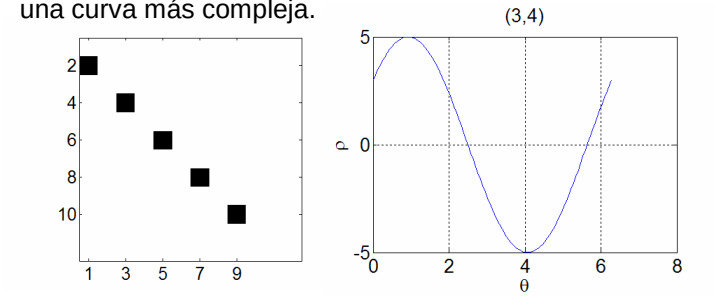


- ORIGEN:
  dado un punto P = (x,y) sobre la recta:

$$\begin{aligned} \vec{n}\vec{P} &= (\cos\theta, \sin\theta)\cdot(x,y) = x\cos\theta + y\sin\theta \\ &= \|\vec{n}\|\|\vec{P}\|\cos\alpha = \rho \end{aligned}$$

Transformada de Hough

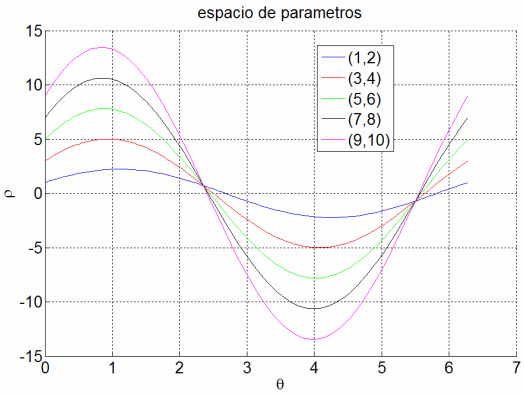
- Ahora los parámetros de la recta son (ρ,θ).
- Diferencias:
  - Ambos parámetros están acotados: ρ ∈ [-D,D] (D = diag(h,v))
    θ ∈ [0,2π]
  - Un píxel de borde no genera una recta de parejas (ρ,θ), sino una curva más compleja.



- No obstante, se aplica el mismo algoritmo que hemos visto.

Transformada de Hough

- Para todos los píxeles de borde:



• El mismo conjunto de puntos da lugar a dos intersecciones que se corresponden con las rectas (ρ,θ) y (–ρ, θ+π).

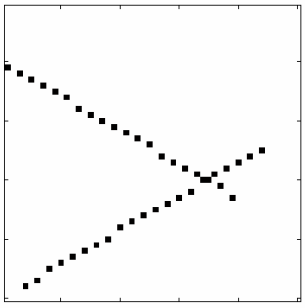
• Si volvemos a la ecuación normal de una recta tenemos:

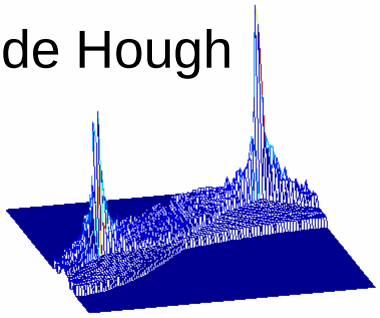
$$\vec{n}\vec{P} = (\cos(\theta), \sin(\theta))\vec{P} = \rho$$
$$\vec{n}_{(+\pi)}\vec{P} = (\cos(\theta+\pi), \sin(\theta+\pi))\vec{P}$$
$$= (-\cos(\theta), -\sin(\theta)) = -\rho$$

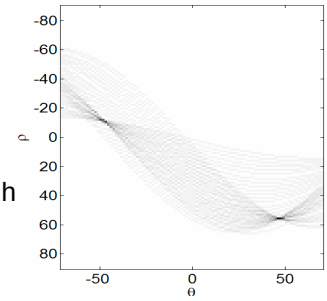
- La segunda intersección se elimina haciendo θ ∈ [–π/2,π/2] (de hecho el resto de ángulos generan rectas fuera de la imagen)

Transformada de Hough

- ¿Qué ocurre cuando hay varias rectas en la imagen?







- Cada pico del acumulador de Hough se corresponde con una de las rectas que acumula más votos.

Transformada de Hough

- Comentarios finales:
  - Para encontrar los picos del acumulador de Hough:
    - buscar el pico más alto y poner a 0 las celdas circundantes
    - repetir el proceso para el resto de líneas que se deseen encontrar (N)
  - Los parámetros del algoritmo son:
    - $\Delta\rho$
    - $\Delta\theta$
    - N (o bien un umbral que indica el número mínimo de píxeles de borde que se han de acumular para considerar una recta relevante)

Indice

- Introducción
- Detección de segmentos rectos mediante códigos de cadena
- Transformada de Hough
- Detección de esquinas

Detección de esquinas

- Se considera que un píxel es una esquina cuando:
  - la derivada de la dirección del gradiente supera un umbral en ese punto, y
  - la magnitud del gradiente en ese punto también supera un umbral
- Se han propuesto muchos detectores de esquinas. Veremos unos cuantos.

Detección de esquinas

- Detector de Kitchen y Rosenfeld (1982):

$$KR = \frac{I_{xx} I_y^2 + I_{yy} I_x^2 - 2 I_{xy} I_x I_y}{I_x^2 + I_y^2}$$

  - Cuanto mayor es la magnitud de KR, más caracter de esquina se le asocia al píxel.
- Indicaciones:
  - para obtener las segundas derivadas se convoluciona la imagen derivada con las máscaras ya vistas en el tema de detección de bordes
  - aplicar supresión de no máximos para evitar respuestas múltiples alrededor de la misma esquina
  - para evitar la introducción de un umbral, seleccionar los N píxeles con más carácter de esquina

Detección de esquinas

- Detector de Nagel-Lipschutz (1969):

$$L = \frac{I_{xx} I_{yy} - I_{xy}^2}{(1 + I_x^2 + I_y^2)^2}$$

  - Cambia de signo a ambos lados de la esquina

- Indicación:
  - para obtener las segundas derivadas se convoluciona la imagen derivada con las máscaras ya vistas en el tema de detección de bordes (cuanto mayor es la máscara, mayor deslocalización se produce)
  - para evitar la introducción de un umbral, seleccionar los N píxeles con más carácter de esquina

Detección de esquinas

- Detector de Harris-Plessey (Harris, 1987; Noble, 1988):

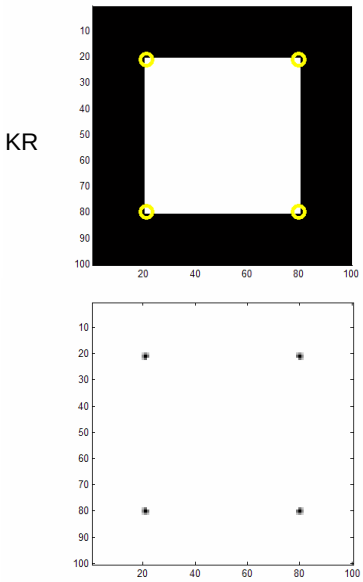
$$H = \frac{\langle I_x^2 \rangle + \langle I_y^2 \rangle}{\langle I_x^2 \rangle \langle I_y^2 \rangle - \langle I_x I_y \rangle^2}$$

  - Cuanto menor es H, más caracter de esquina se le concede al píxel
  - Este detector sólo emplea derivadas de primer orden
  - $\langle \cdot \rangle$ significa convolución con un filtro gaussiano con una cierta σ

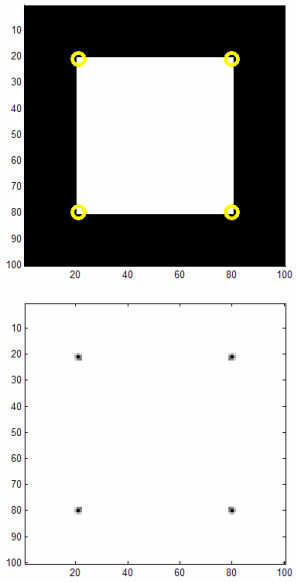
- Indicación:
  - aplicar supresión de no máximos para evitar respuestas múltiples alrededor de la misma esquina
  - para evitar la introducción de un umbral, seleccionar los N píxeles con más carácter de esquina

Detección de esquinas

KR

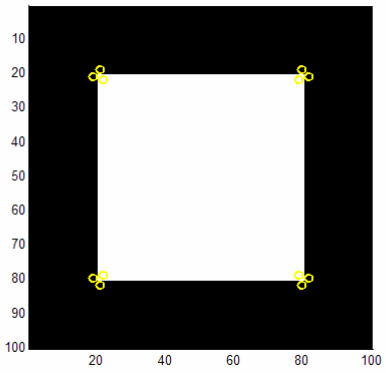


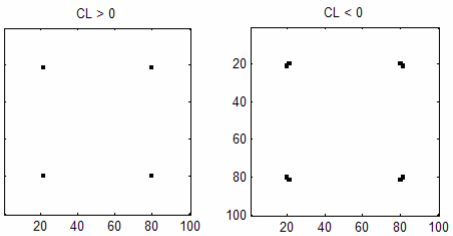
Harris



Detección de esquinas

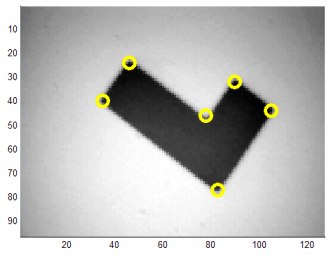
Lipschutz

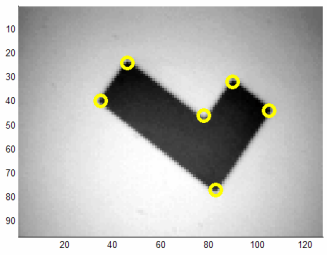




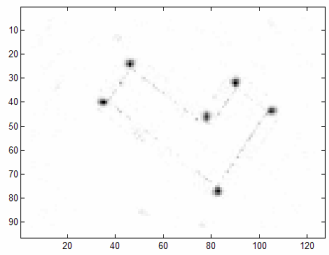
Detección de esquinas

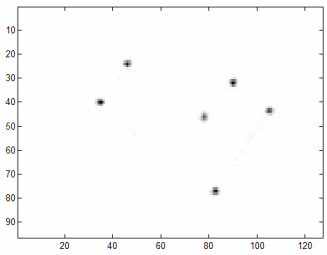
KR





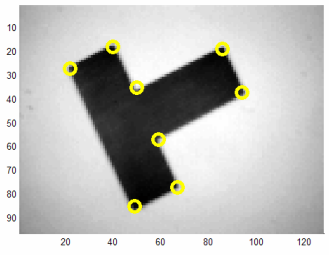
Harris

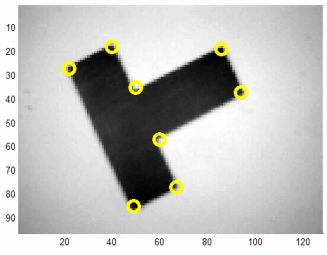




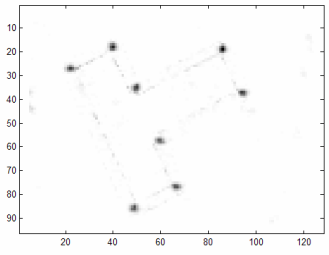
Detección de esquinas

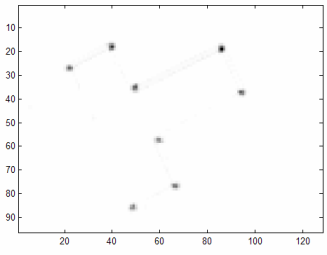
KR





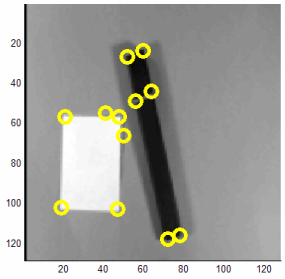
Harris

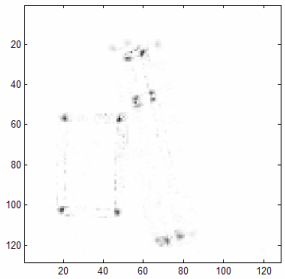




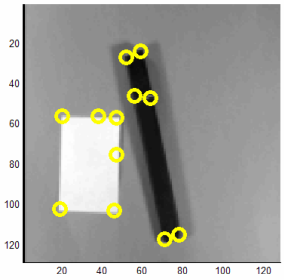
Detección de esquinas

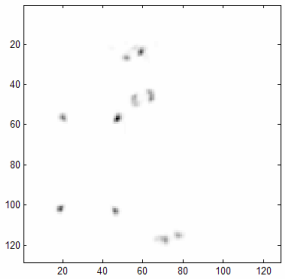
KR





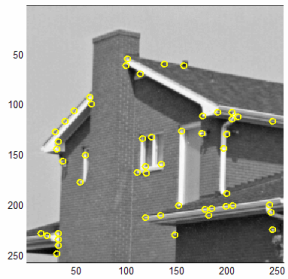
Harris

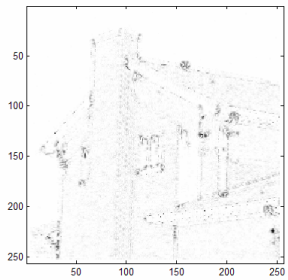




Detección de esquinas

KR





Harris