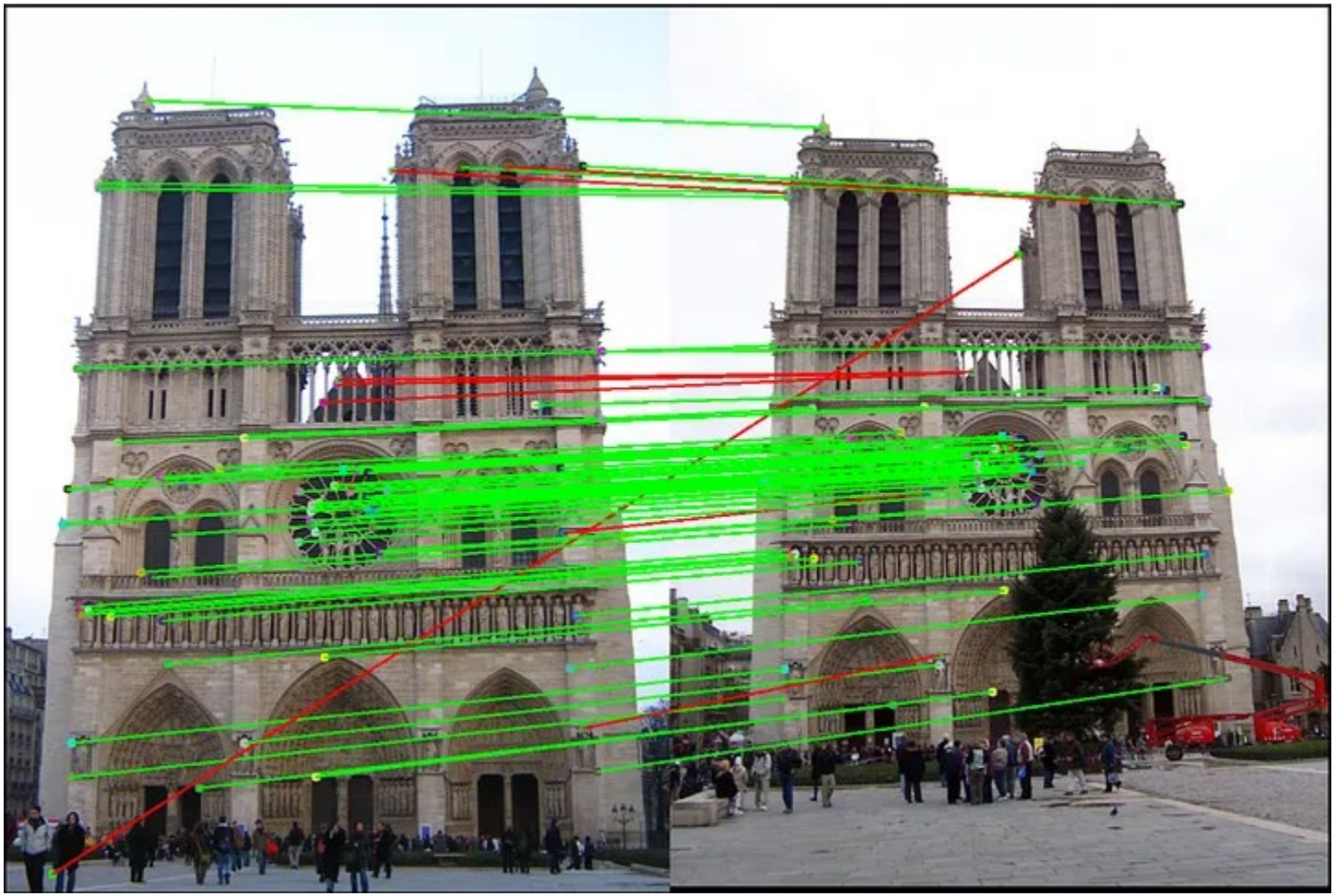
# Visión por Computador

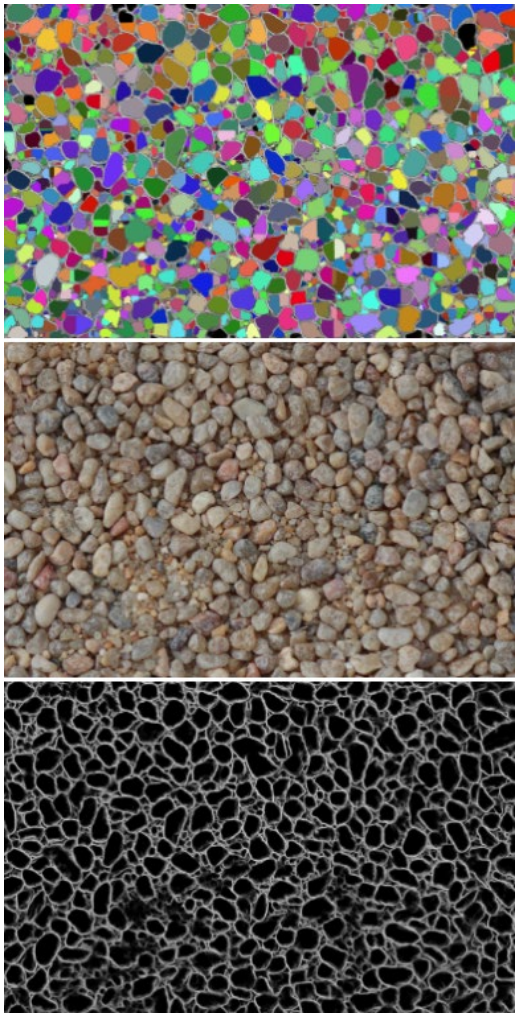
## Tema 4. Detección de características



EII, GII. *Modesto Castrillón Santana*

# Contenidos

- Introducción
- Puntos
- Bordes
- Líneas
- Segmentación

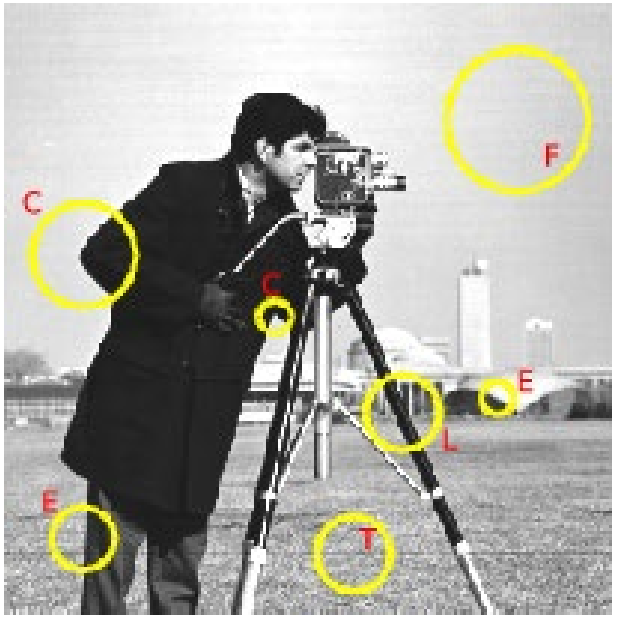


# Introducción

Detección de características y encaje (*matching*)

Fundamental en múltiples aplicaciones VC

Puntos, esquinas, líneas, contornos y parches (*patches*)

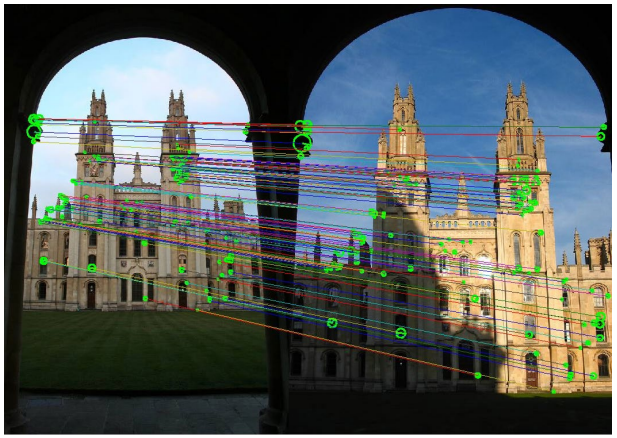


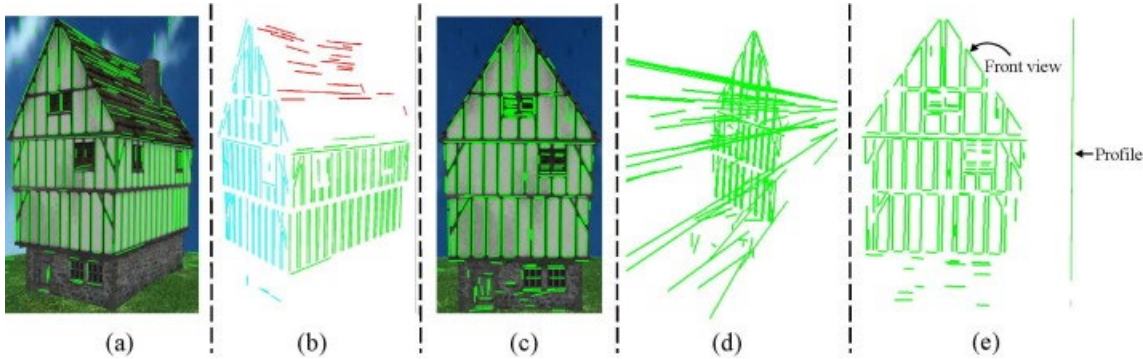
# Aplicaciones

## Detección de elementos y formas

# Aplicaciones

## Alineamiento/encaje de imágenes desde puntos o líneas





# Aplicaciones

## Alineamiento/encaje de imágenes desde puntos o líneas

## ¿Qué es un punto de interés?

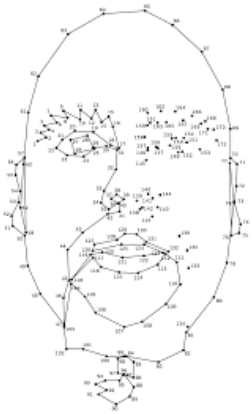


Imagen original o preprocesada, p.e. bordes

Presencia de elementos singulares/característicos:

- Extremos
- Cambios de dirección

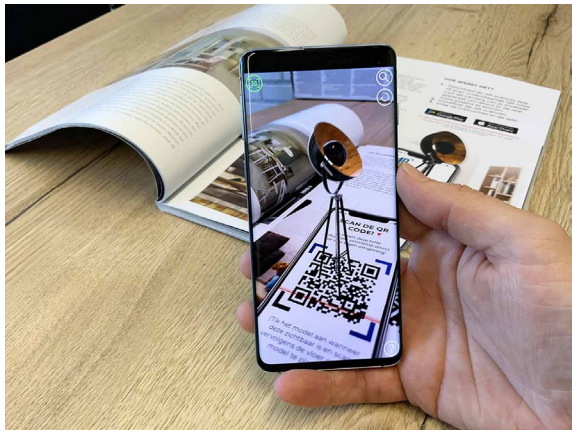
# ¿Qué es un punto de interés?

Correspondencia entre imágenes
¿Es el mismo objeto?

# Aplicaciones

Alineamiento de imágenes:

- Calibración de cámara
- Encaje estéreo
- Reconocimiento de patrones
- *Stitching* de imágenes (mosaicos)
- Consulta de imágenes (*image retrieval*)
- Seguimiento
- Estabilización de vídeo (cámaras, AR)



# Buen punto de interés

¿Cuál(es) es/son más útil(es) para la correspondencia entre imágenes?

# Buen punto de interés

¿Cuál(es) es/son más útil(es) para la correspondencia entre imágenes?

Distintivo y repetible

# Buen punto de interés

Correspondencia de puntos:

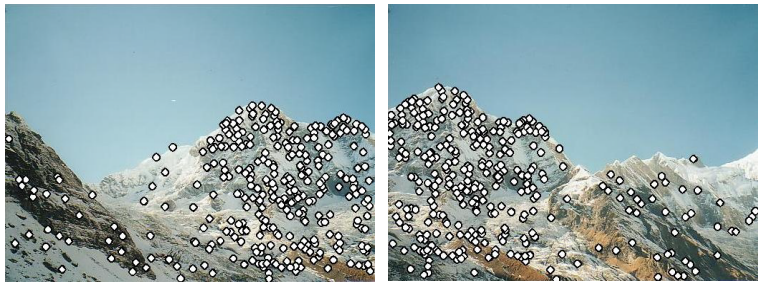
- Detectar en una imagen
  Vídeo, elementos próximos, imágenes sucesivas
  Encaje por correlación en zonas cercanas (seguimiento)

- Detectar en imágenes, encajar en base a apariencia local
  Más general

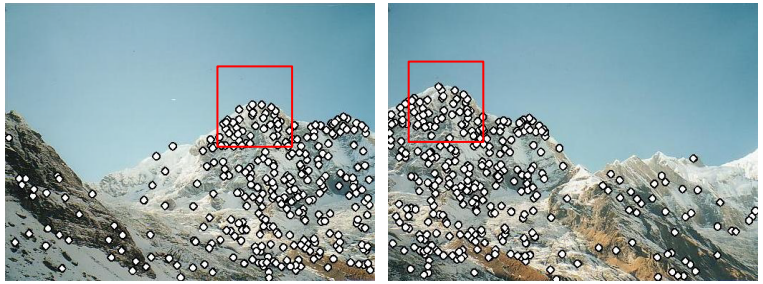
# Proceso

Etapas:

- Detección
  Localizar puntos de interés

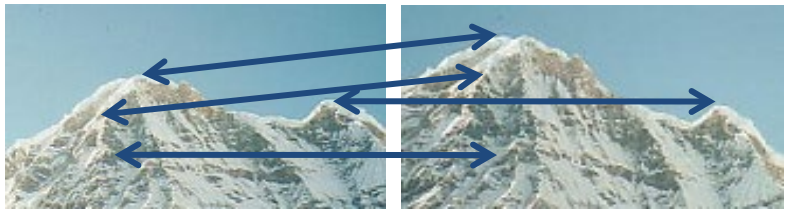


- Descripción
  Caracterizar puntos de interés



$$x_1 = [x_1^1, x_2^1, \dots, x_d^1] \qquad x_2 = [x_1^2, x_2^2, \dots, x_d^2]$$

- Encaje
  Determinar la correspondencia

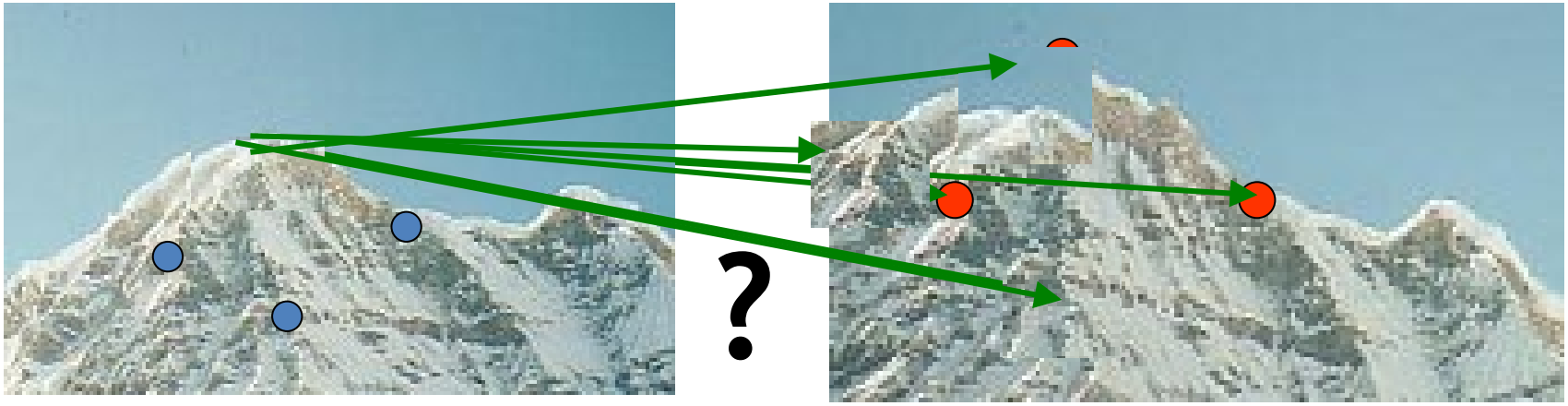


Szeliski

# Etapas

Determinar correspondencia de puntos en varias imágenes

Dificultad en contextos con repetición de elementos

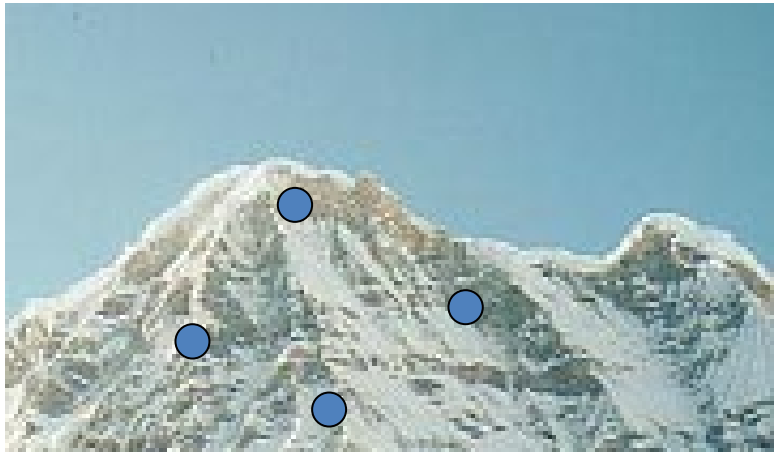


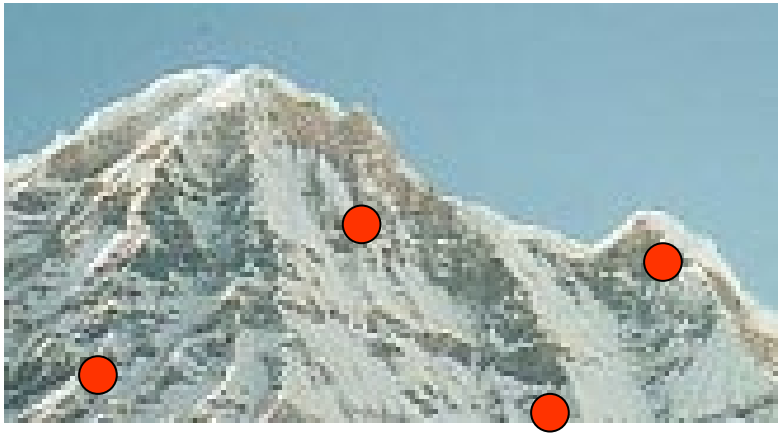

Szeliski

# Detección

Características:

- **Distintivos**: Cada elemento único
- **Repetible**: Localizables en múltiples imágenes, aún con variaciones geométricas y fotométricas
- **Compacidad**: Descripción reducida << que la imagen
- **Eficiencia**: Al comparar/encajar



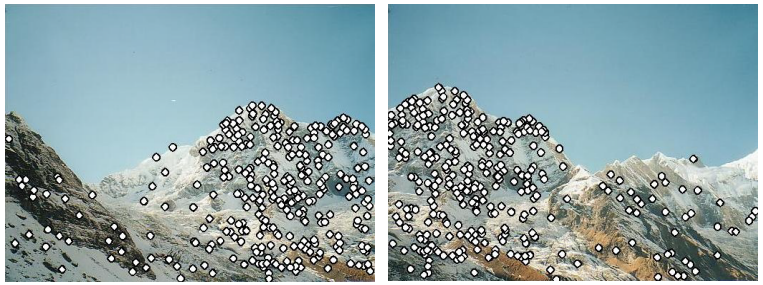


Szeliski

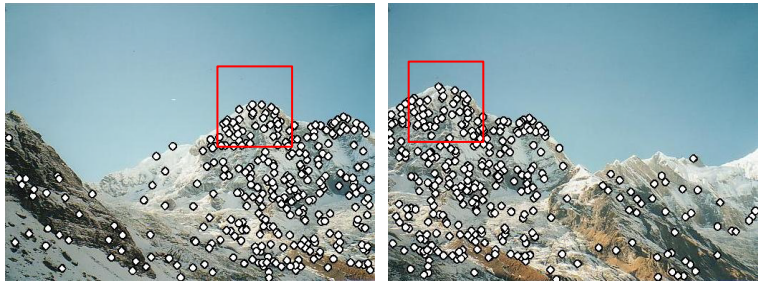
# Proceso

Etapas:

- **Detección**
  Localizar puntos de interés

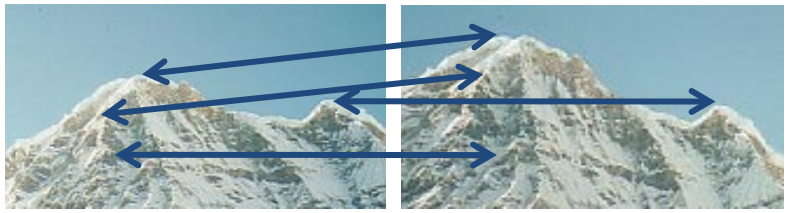


- Descripción
  Caracterizar puntos de interés



$$x_1 = [x_1^1, x_2^1, \dots, x_d^1] \qquad x_2 = [x_1^2, x_2^2, \dots, x_d^2]$$

- Encaje
  Determinar la correspondencia



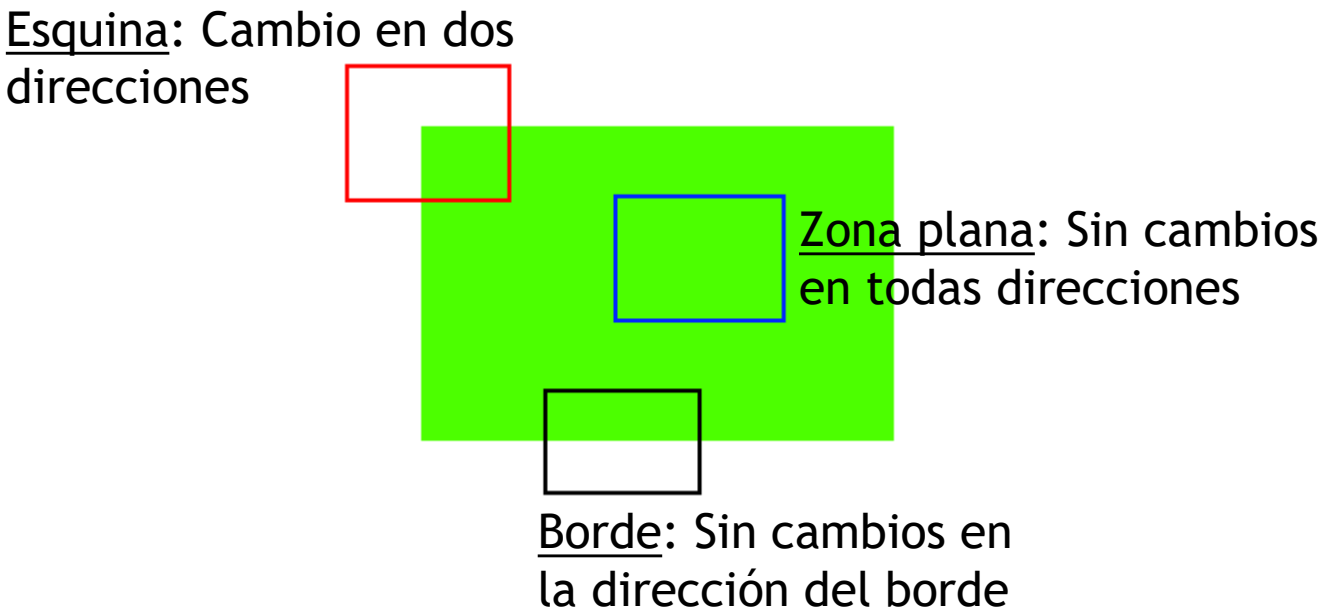
Szeliski

# Detección

Un punto de interés presenta cambios en su entorno

Variaciones de intensidad en la vecindad

Estabilidad local frente a pequeños desplazamientos

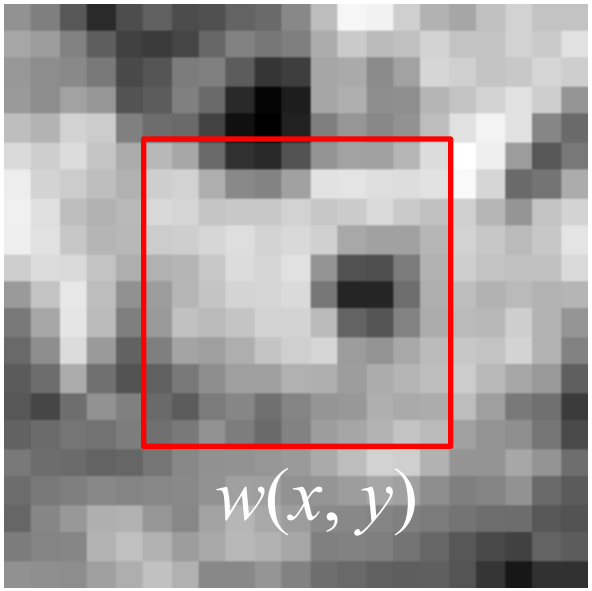


# Detección

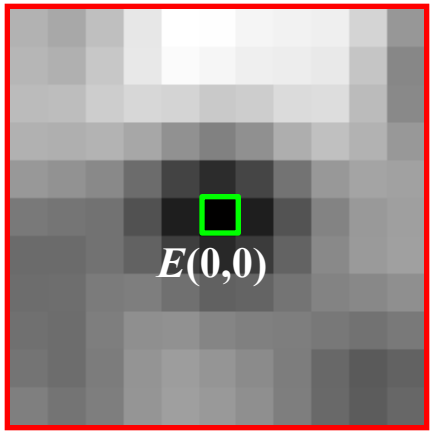
Autocorrelación en ventana *w* con desplazamiento *(u,v)*
Suma de las diferencias entre dos trozos de imagen

$$E(u,v) = \sum_{x,y} w(x,y)[f(x+u, y+v) - f(x,y)]^2$$

$f(x, y)$




$E(u, v)$



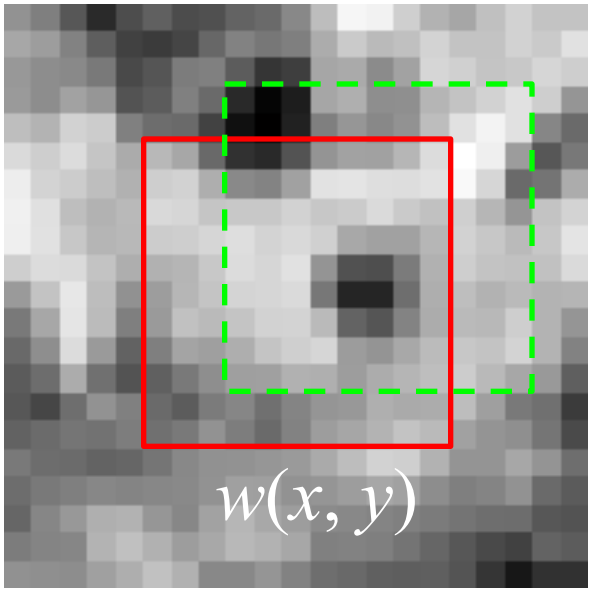

Negro, mínimo

# Detección

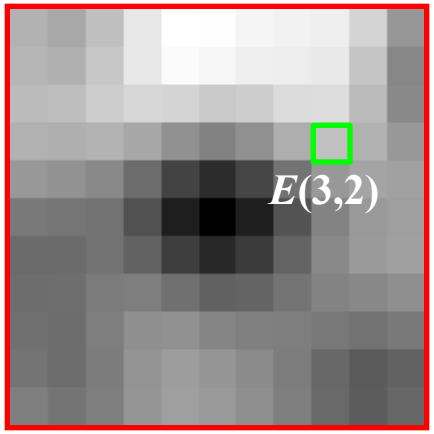
Autocorrelación en ventana *w* con desplazamiento *(u,v)*
Suma de las diferencias entre dos trozos de imagen

$$E(u,v) = \sum_{x,y} w(x,y)[f(x+u, y+v) - f(x,y)]^2$$

$f(x, y)$




$E(u, v)$




Negro, mínimo

Szeliski

# Detección

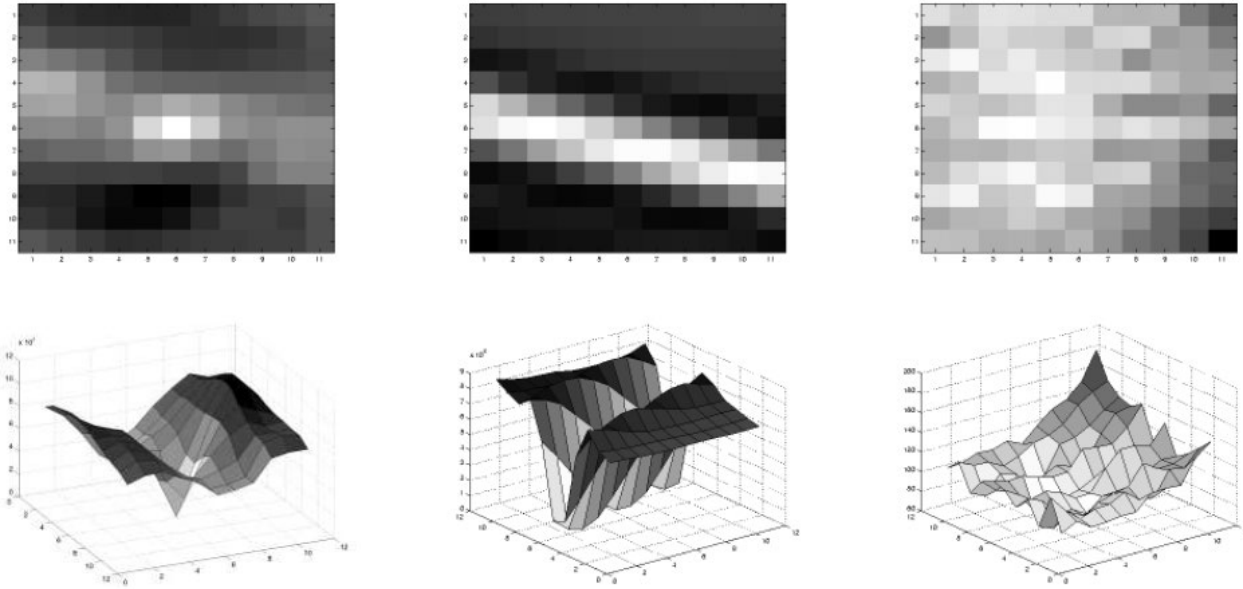
Salida, matriz 2D (superficie)

$$E(u,v) = \sum_{x,y} w(x,y)[f(x+u, y+v) - f(x,y)]^2$$

Szeliski

Puntos

Ojo, blanco es mínimo



(b) (c) (d)

Szeliski

# Buenos puntos

Conocida como SSD (*sum of squared differences*)

$$E(u,v) = \sum_{x,y} w(x,y)[f(x+u, y+v) - f(x,y)]^2$$

Coste elevado

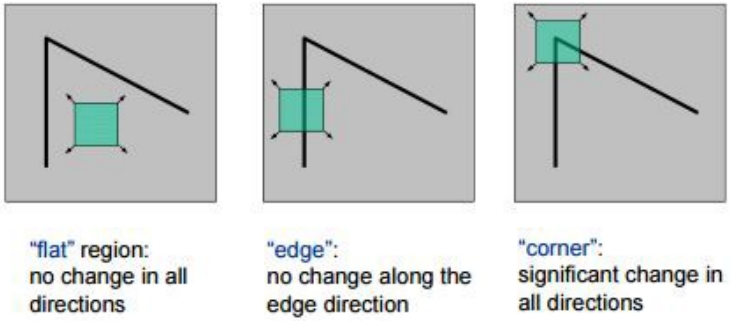
*o(ancho_w$^2$ * rango_desp$^2$ * ancho_img$^2$)*

# Buenos puntos

Esquina: gradiente local con dos direcciones dominantes

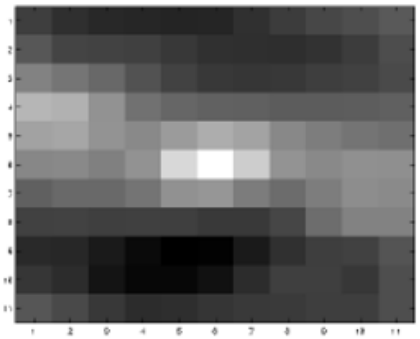
Esquinas repetibles y distintivas

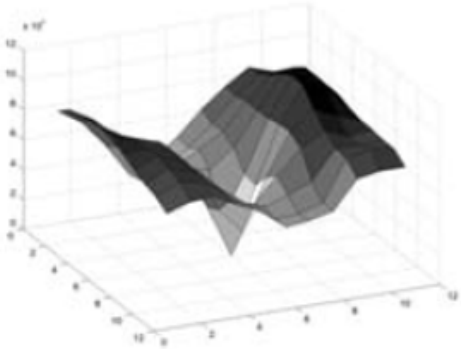
Estabilidad apariencia local




Busca poca estabilidad de *E(u,v)*

Localiza mínimo(s) local(es) marcado(s) en *E(u,v)* (blanco en gráfica)





Szeliski

# Harris

Técnica para localizar esquinas (1988)

Explora vecindad del píxel

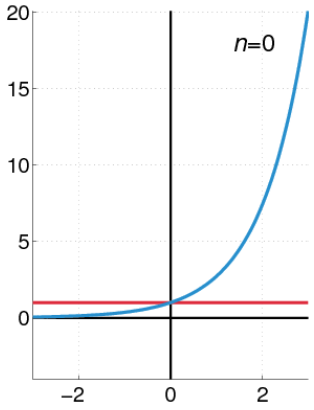
$$E(u,v) = \sum_{x,y} w(x,y)[f(x+u, y+v) - f(x,y)]^2$$

Interesan *E(u,v)* con diferencias (segundo término)

Alto coste, aproxima *E(u,v)*

C.Harris and M.Stephens. "A Combined Corner and Edge Detector" *4th Alvey Vision Conference*:

# Harris



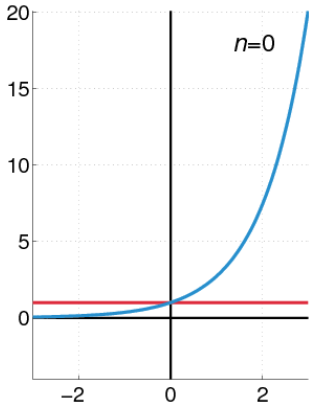

Aproximación local superficie SSD

Expansión con serie de Taylor

Función como serie infinita de sus derivadas en un punto

$$f(a) = \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

# Harris




Aproximación local de la superficie SSD

Expansión con serie de Taylor

Función como serie infinita de sus derivadas en un punto

$$f(a) = \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

*E(u,v)* cerca de (0,0)

$$E(u,v) \approx E(0,0) + [u \quad v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \quad v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{vu}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

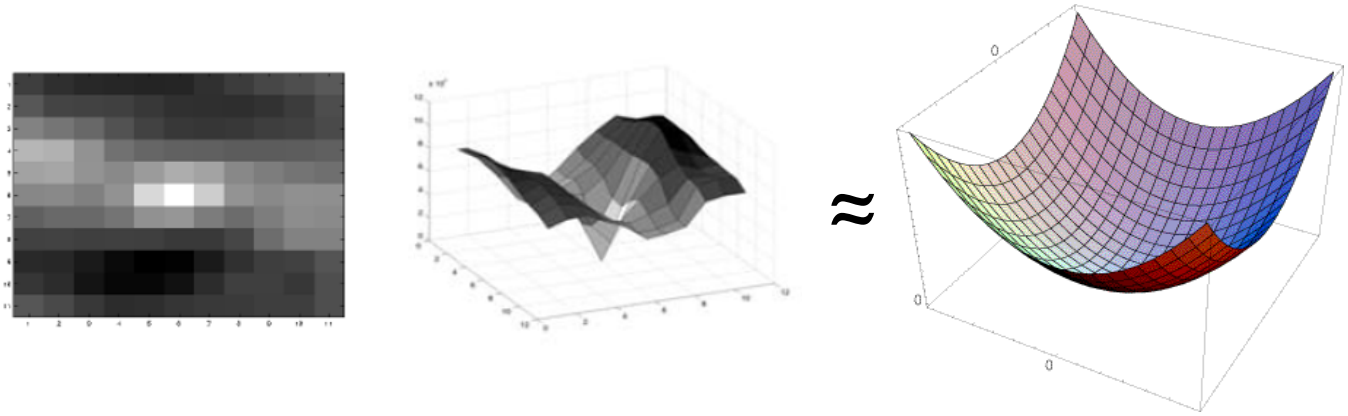
# Harris

Expansión con serie de Taylor

Asumiendo $E(0,0)=0$, $E_u(0,0)=0$ y $E_v(0,0)=0$

$$E(u,v) \approx (u \quad v) M \begin{pmatrix} u \\ v \end{pmatrix} \qquad M = \sum_{x,y} w(x,y) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$$

Segundo orden de la serie, una superficie cuadrática

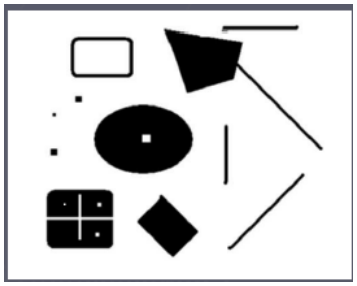


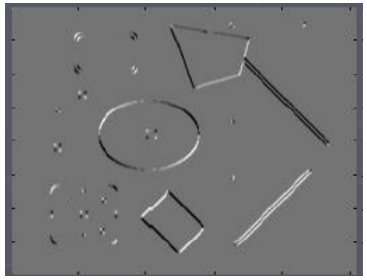
James Hays

# Harris

$$M = \sum_{x,y} w(x,y) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$$

Gradientes o diferencias de intensidades en imagen





Donde: $I_x \Leftrightarrow \frac{\partial I}{\partial x}$ $I_y \Leftrightarrow \frac{\partial I}{\partial y}$ $I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$

James Hays

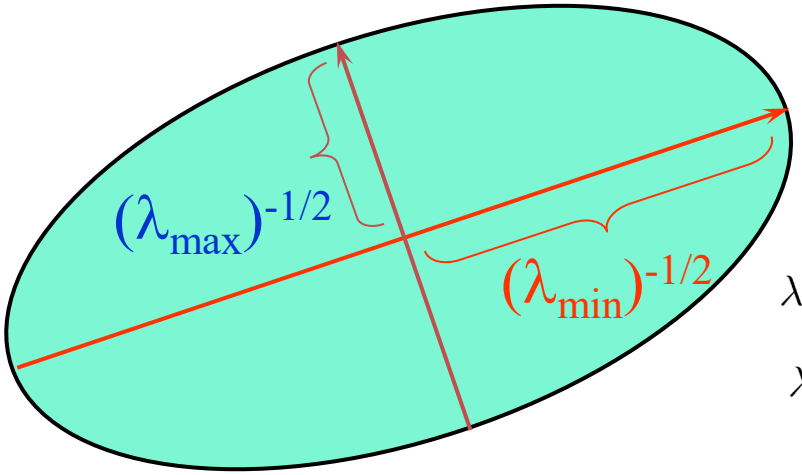
# Harris

*M* es:

- Simétrica, autovectores ortogonales
- Cuadrada, puede ser diagonalizable

$$M = \begin{pmatrix} I_x I_x & I_x I_y \\ I_y I_x & I_y I_y \end{pmatrix} = C^{-1} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} C$$

- Los autovalores ($\lambda_1$ y $\lambda_2$) describen una elipse




$$\lambda_{max} = \frac{1}{a^2} \implies a = (\lambda_{max})^{-1/2}$$

$$\lambda_{min} = \frac{1}{b^2} \implies b = (\lambda_{min})^{-1/2}$$

James Hays

# Harris

$$M = \begin{pmatrix} I_x I_x & I_x I_y \\ I_y I_x & I_y I_y \end{pmatrix} = C^{-1} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} C$$

En una matriz diagonal:

$$\det(M) = \prod_{i=1}^{n} \lambda_i \qquad \text{traza}(M) = \sum_{i=1}^{n} \lambda_i$$

Obtener *det* y *traza* sin diagonalizar

**Criterio de decisión** en base al valor de S:

$$S = \det(M) - k\big(\text{traza}(M)\big)^2 \text{ con } k = 0.04\text{-}0.06$$

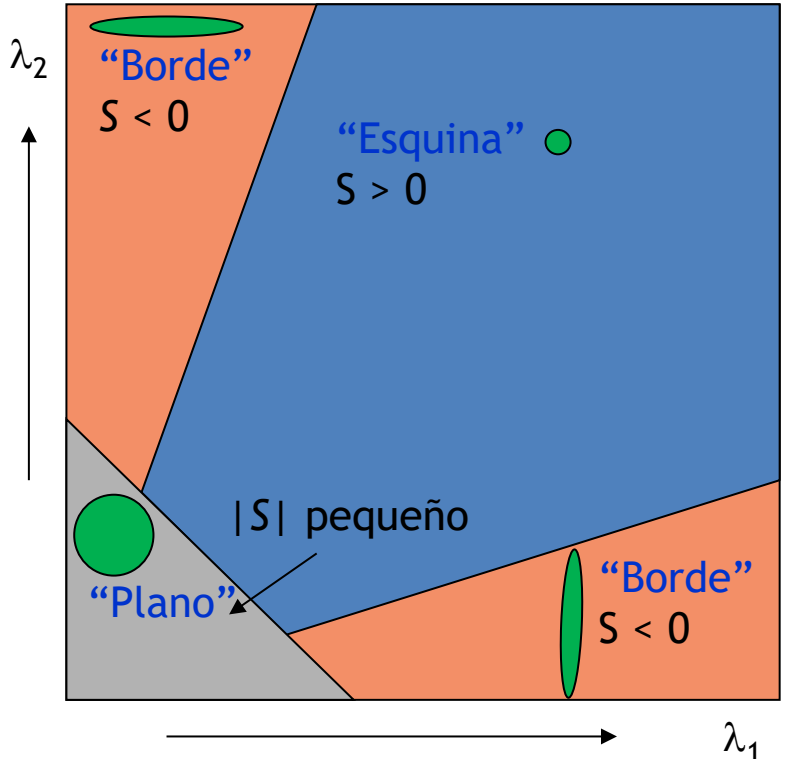
*k* empírico, 0.04 más sensible, 0.06 menos sensible

# Harris

$$S = \det(M) - k\left(traza(M)\right)^2$$

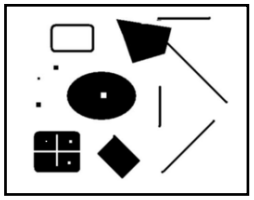
Si:

- |S| es pequeño, $\lambda_1$ y $\lambda_2$ pequeños: zona plana
- S<0, $\lambda_1 >> \lambda_2$ o viceversa: borde
- S>0, $\lambda_1$ y $\lambda_2$ grandes y similares $\lambda_1 \sim \lambda_2$: esquina



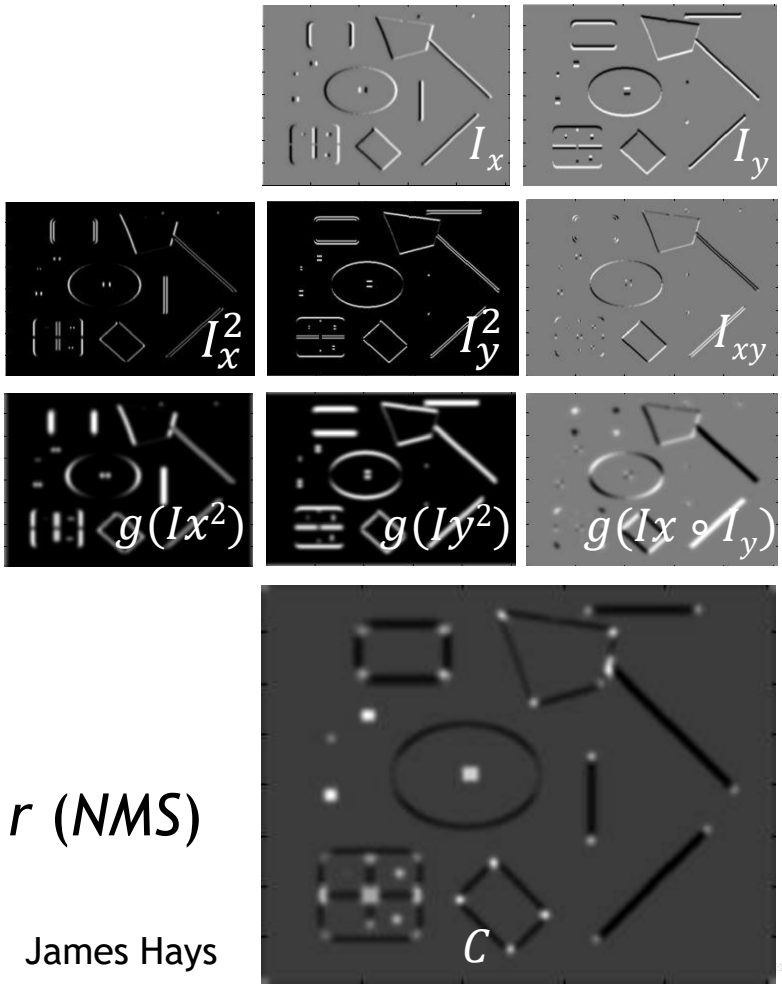

Adaptado de Szeliski

# Harris



Cálculo de M en cada píxel:

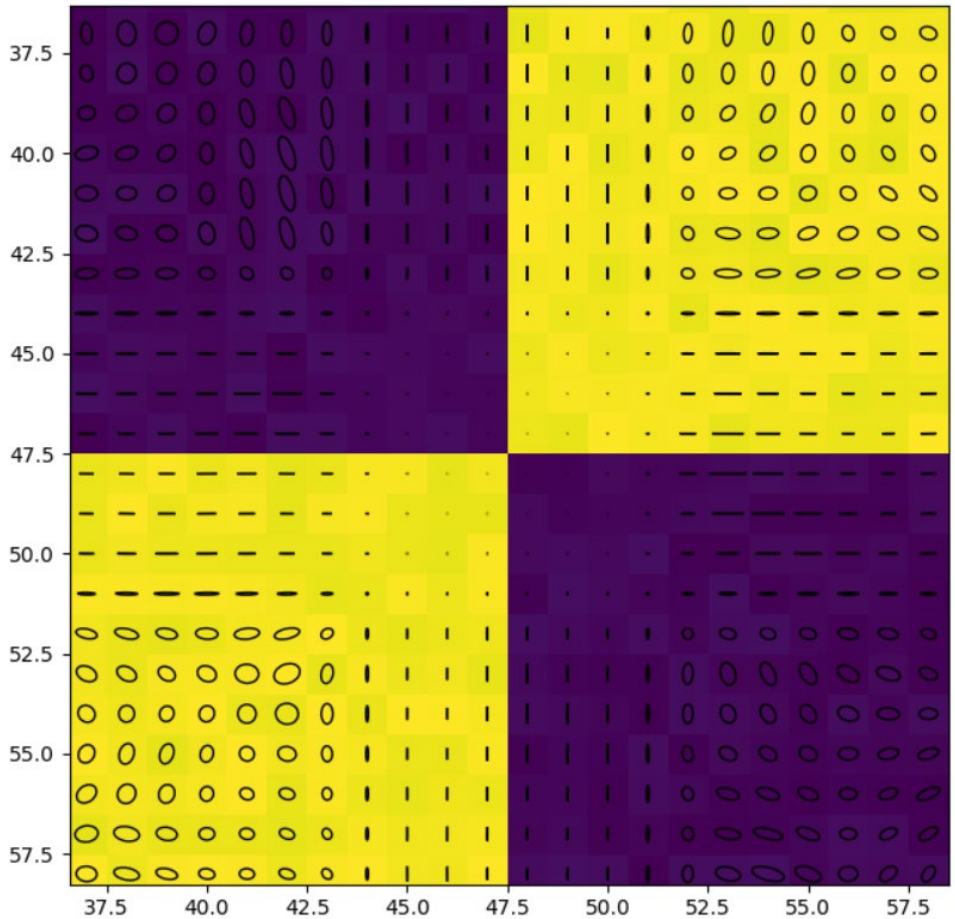
- Gaussiana
- Gradientes $I_x$ e $I_y$
- Componentes de $M$
- Calcular $S$
- Desechar valores pequeños (≈0)
- Eliminar redundancias en cierto $r$ ($NMS$)




James Hays

# Harris

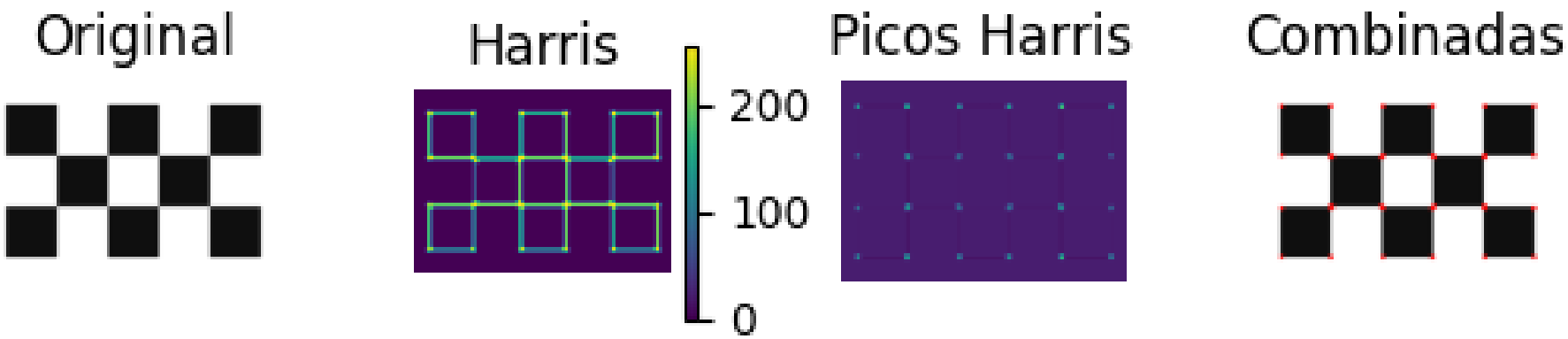
$$S = \det(M) - k\left(traza(M)\right)^2$$



# Harris

## Resultado

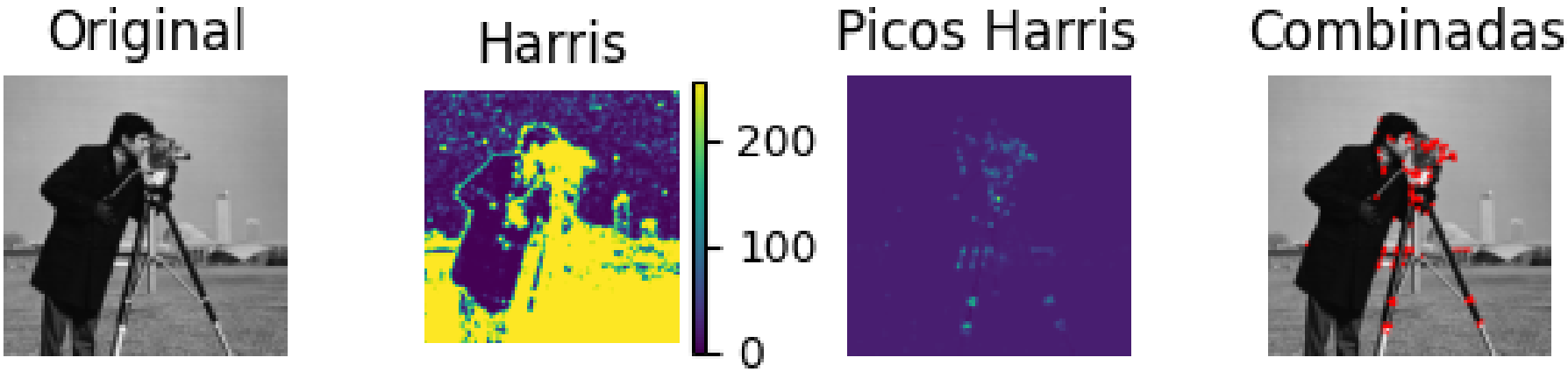
```
gris = np.float32(gris) #Opera con 32 bits
dst = cv2.cornerHarris(gris,2,3,0.04) #Calcula Harris
dst = cv2.dilate(dst, None) #Dilata para resaltar
cth=0.1
rgb[dst>cth*dst.max()]=[255,0,0] #Remarca valores altos
```



# Harris

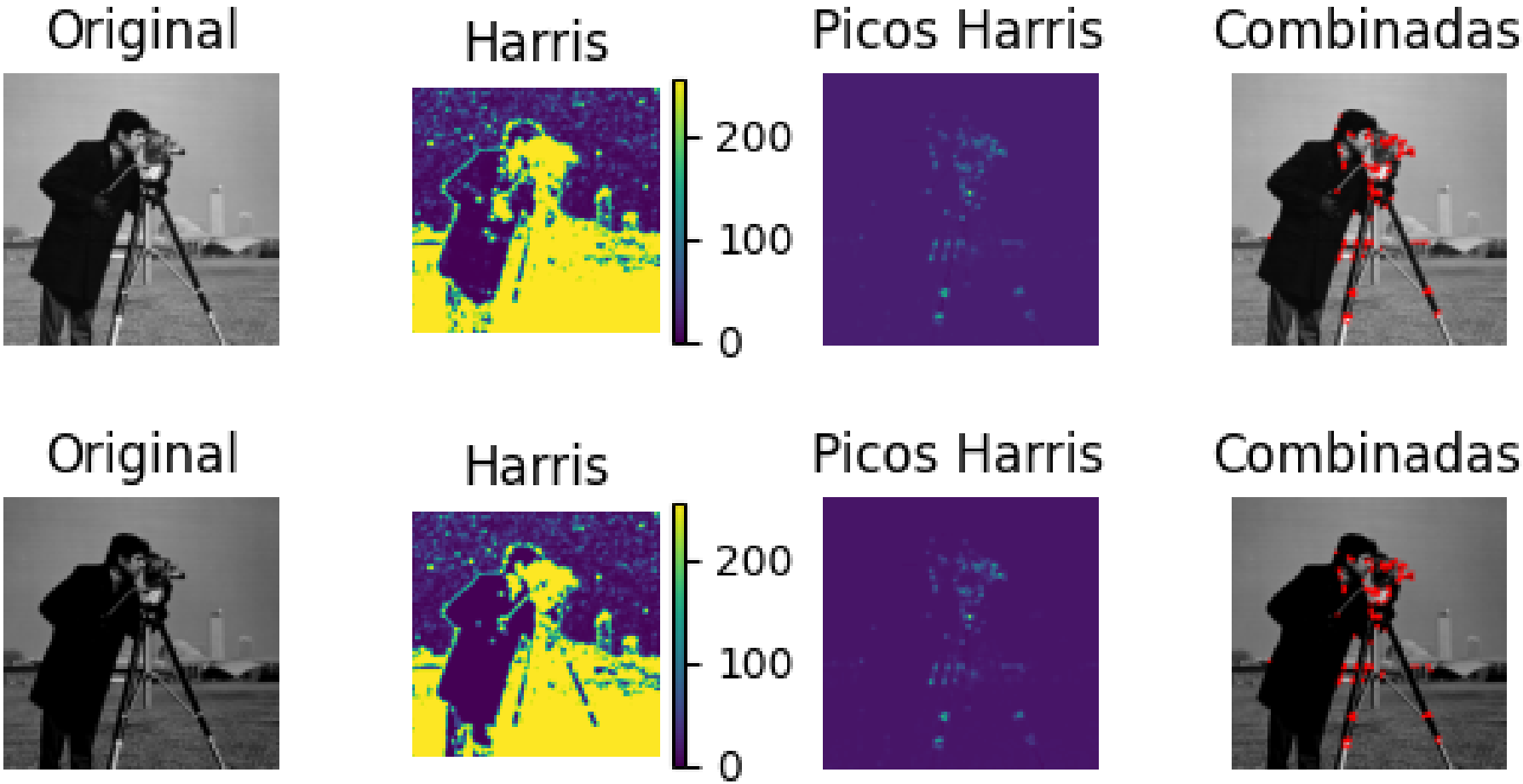
## Resultado

```
gris = np.float32(gris) #Opera con 32 bits
dst = cv2.cornerHarris(gris,2,3,0.04) #Calcula Harris
dst = cv2.dilate(dst, None) #Dilata para resaltar
cth=0.1
rgb[dst>cth*dst.max()]=[255,0,0] #Remarca valores altos
```



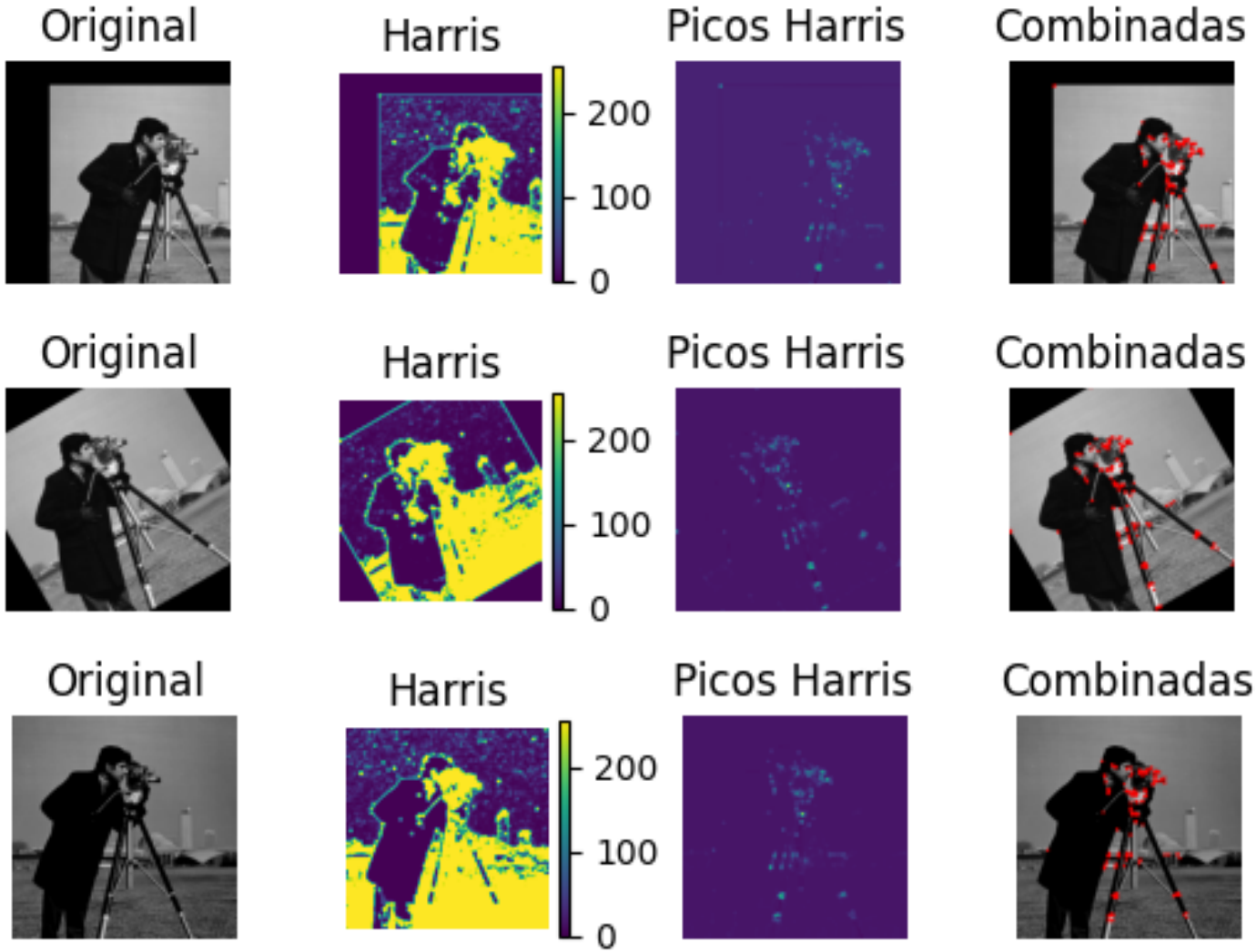
# Harris

Invarianza a traslaciones, rotaciones y cambios de brillo



# Harris

Invarianza a traslaciones, rotaciones y cambios de brillo



# Harris

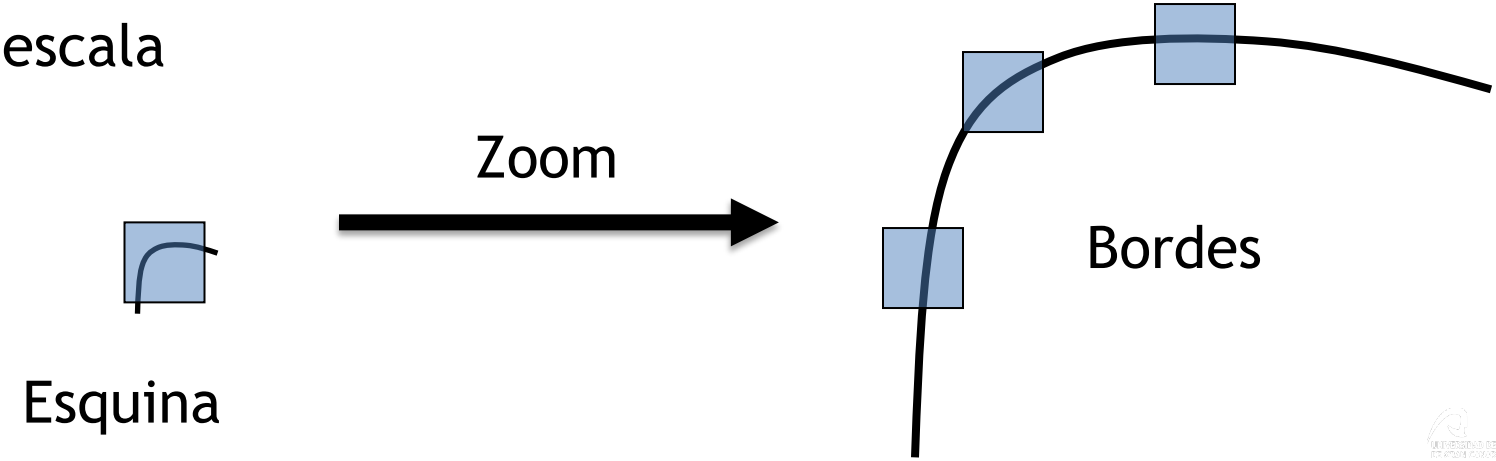
Invariante

Cálculo dependiente de gradientes/derivadas

No afectadas por:

- Traslaciones
- Rotaciones
- Iluminosidad

Sí por escala



# Harris

¿Invarianza a escala?

Calcular a múltiples resoluciones (pirámide)

Extraer características estables a escala y posición

LoG se adapta bien a múltiples resoluciones (Harris-Laplace)

*Difference of Gaussians* (DoG) como aproximación de LoG

Histogramas de gradientes orientados locales (SIFT)

*Maximally Stable Extremal Region* (MSER)

# Comparativa

Table 7.1 Overview of feature detectors.

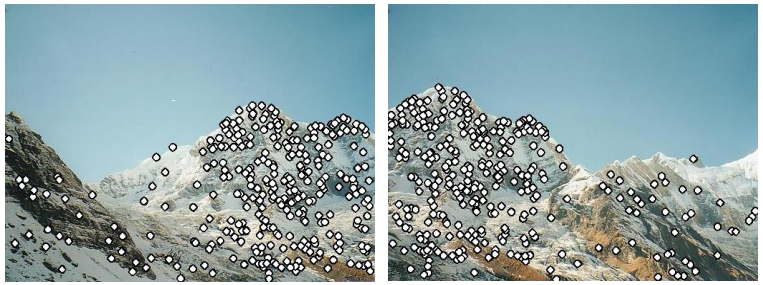
| Feature Detector | Corner | Blob | Region | Rotation invariant | Scale invariant | Affine invariant | Localization Repeatability | Localization accuracy | Robustness | Efficiency |
|---|---|---|---|---|---|---|---|---|---|---|
| Harris | √ | | | √ | | | +++ | +++ | +++ | ++ |
| Hessian | | √ | | √ | | | ++ | ++ | ++ | + |
| SUSAN | √ | | | √ | | | ++ | ++ | ++ | +++ |
| Harris-Laplace | √ | (√) | | √ | √ | | +++ | +++ | ++ | + |
| Hessian-Laplace | (√) | √ | | √ | √ | | +++ | +++ | +++ | + |
| DoG | (√) | √ | | √ | √ | | ++ | ++ | ++ | ++ |
| SURF | (√) | √ | | √ | √ | | ++ | ++ | ++ | +++ |
| Harris-Affine | √ | (√) | | √ | √ | √ | +++ | +++ | ++ | ++ |
| Hessian-Affine | (√) | √ | | √ | √ | √ | +++ | +++ | +++ | ++ |
| Salient Regions | (√) | √ | | √ | √ | (√) | + | + | ++ | + |
| Edge-based | √ | | | √ | √ | √ | +++ | +++ | + | + |
| MSER | | | √ | √ | √ | √ | +++ | +++ | ++ | +++ |
| Intensity-based | | | √ | √ | √ | √ | ++ | ++ | ++ | ++ |
| Superpixels | | | √ | √ | (√) | (√) | + | + | + | + |

Tuytelaars y Mikolajczyk 2008

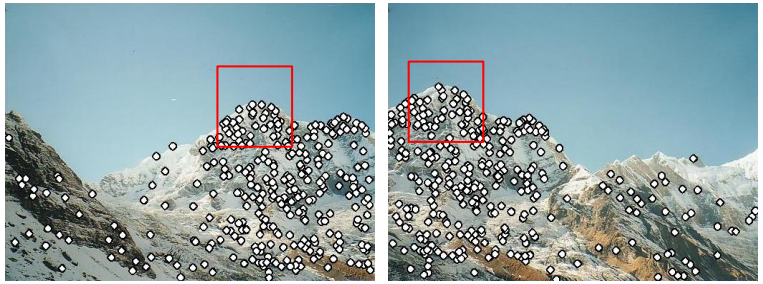
# Proceso

Etapas:

- Detección
  Localizar puntos de interés

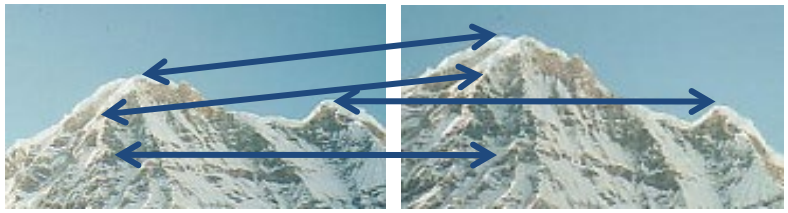


- **Descripción**
  Caracterizar puntos de interés



$$x_1 = [x_1^1, x_2^1, \dots, x_d^1] \qquad x_2 = [x_1^2, x_2^2, \dots, x_d^2]$$

- Encaje
  Determinar la correspondencia



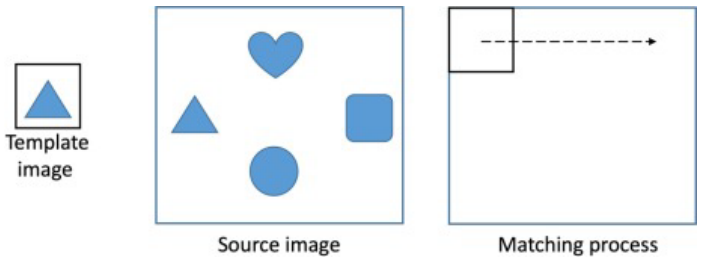
Szeliski

# Descripción

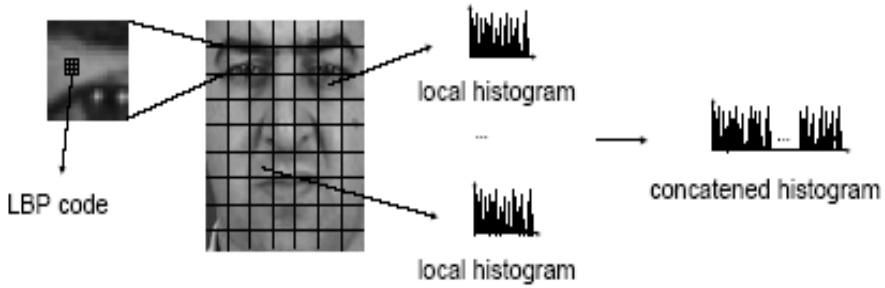
Tras localizar puntos singulares
Describir para encajarlos

Posibilidades de descripción:

- Plantilla: Tolera traslaciones
  SSD o similar




- Histograma: Tolera también cambios de escala y orientación
  Color, textura, SIFT, etc.
  Distancia de histogramas

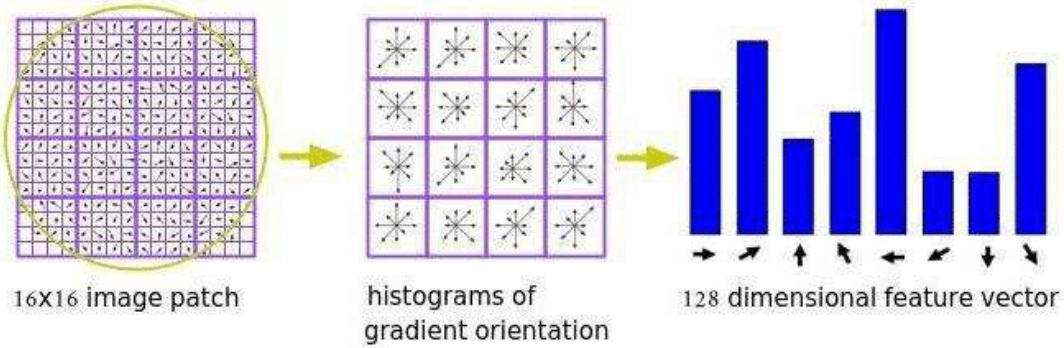


# Descripción

Descriptores (alternativas según aplicación):

- *Binary Robust Independent Elementary Features* (BRIEF)
- *A Fast Local Descriptor for Dense Matching* (DAISY)
- *Gradient location-orientation histogram* (GLOH)
- *Scale Invariant feature transform* (SIFT patente expiró en 2020)
- PCA-SIFT
- RootSIFT
- *Bias and gain normalization* (MOPS)
- Oriented FAST and Rotated BRIEF (ORB)
- *Speeded-Up Robust Features* (SURF)
- …

Ilustración con SIFT:



# SIFT

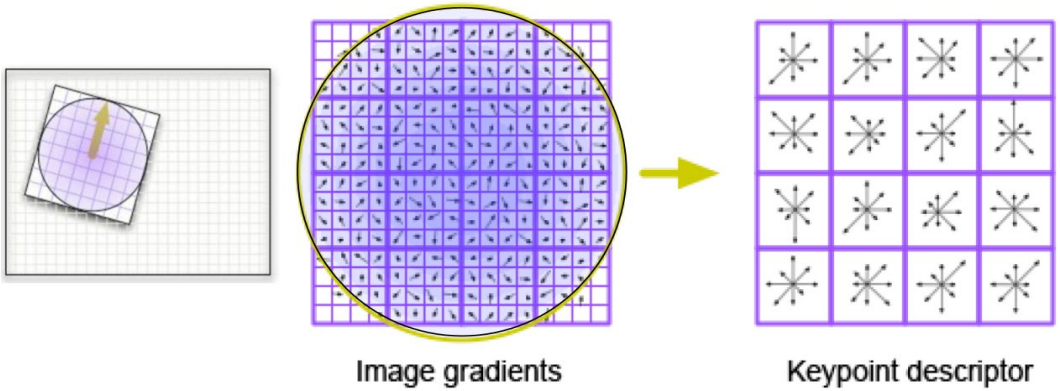
Invariante a cambios de escala (D. Lowe, 2004)

Textura como distribución de diferencias de la imagen

Orientación gradiente dominante en la vecindad

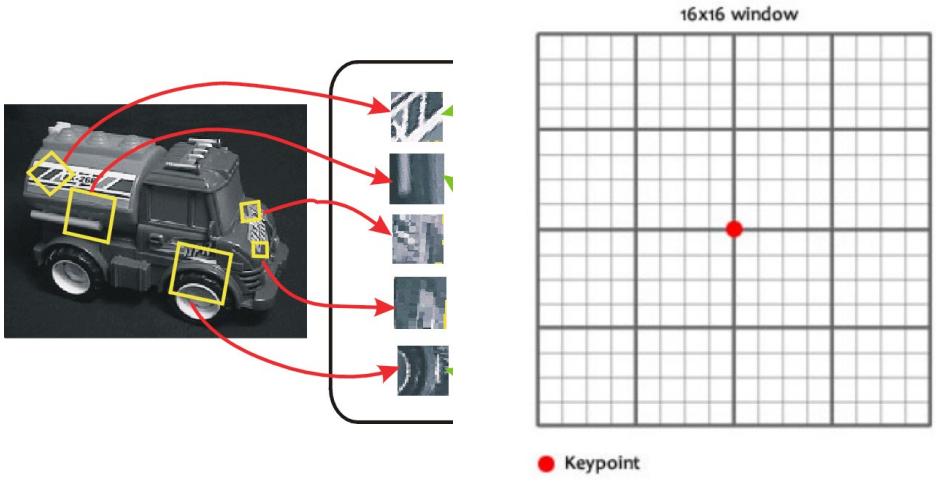
Cálculo del punto de interés en pirámide

Histograma local de gradientes orientados



# SIFT

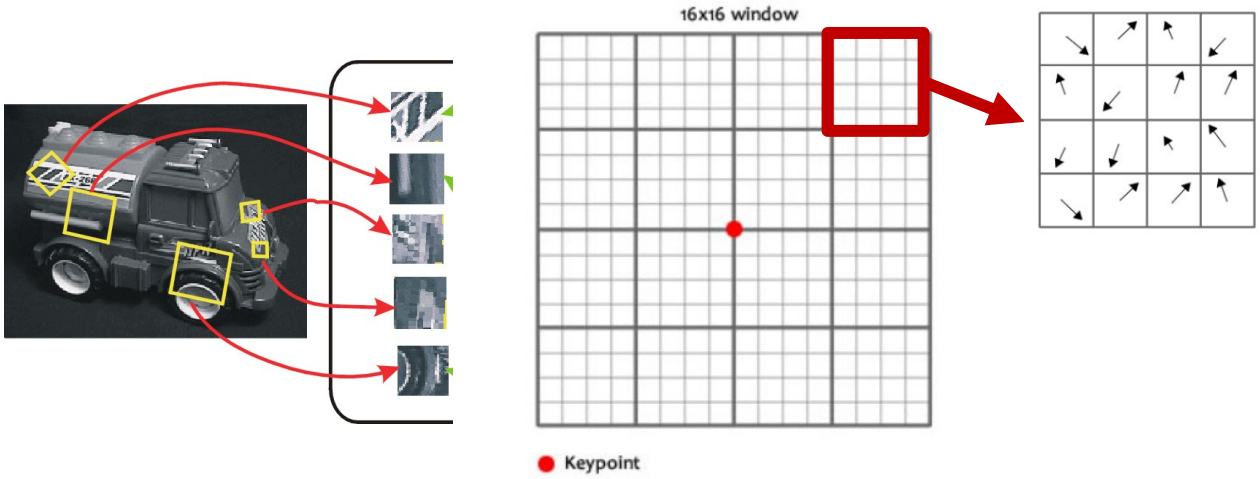
Ventana 16x16 (amarilla) en puntos de interés (rojos)




Adaptado de James Hays y Utkarsh Sinha

# SIFT

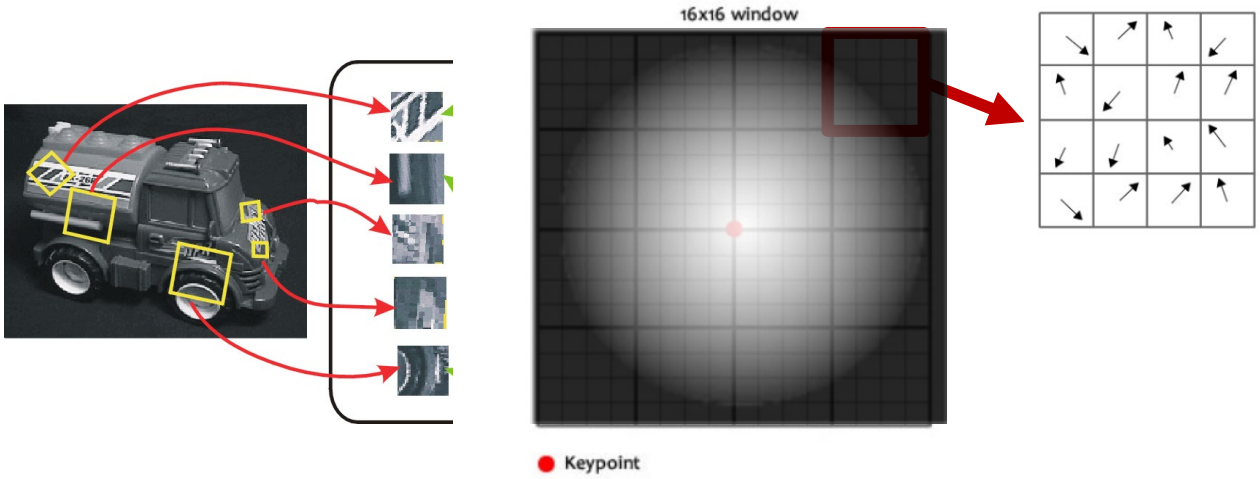
Ventana 16x16 (amarilla) en puntos de interés (rojos)
Magnitud y orientación del gradiente por píxel




Adaptado de James Hays y Utkarsh Sinha

# SIFT

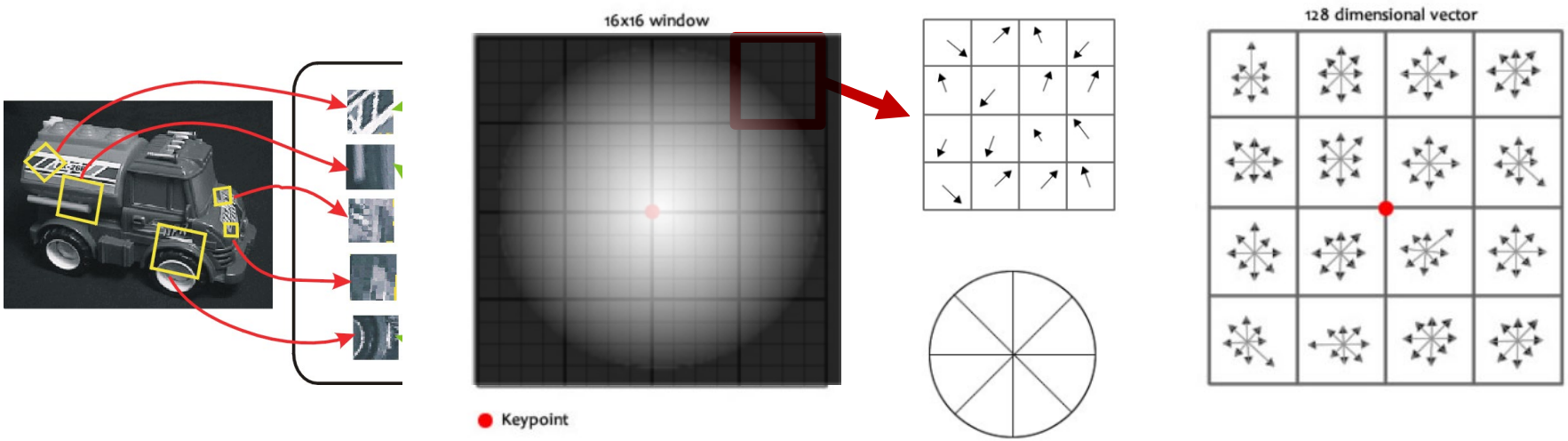
Ventana 16x16 (amarilla) en puntos de interés (rojos)
Magnitud y orientación del gradiente por píxel
Ponderado con Gaussiana




Adaptado de James Hays y Utkarsh Sinha

# SIFT

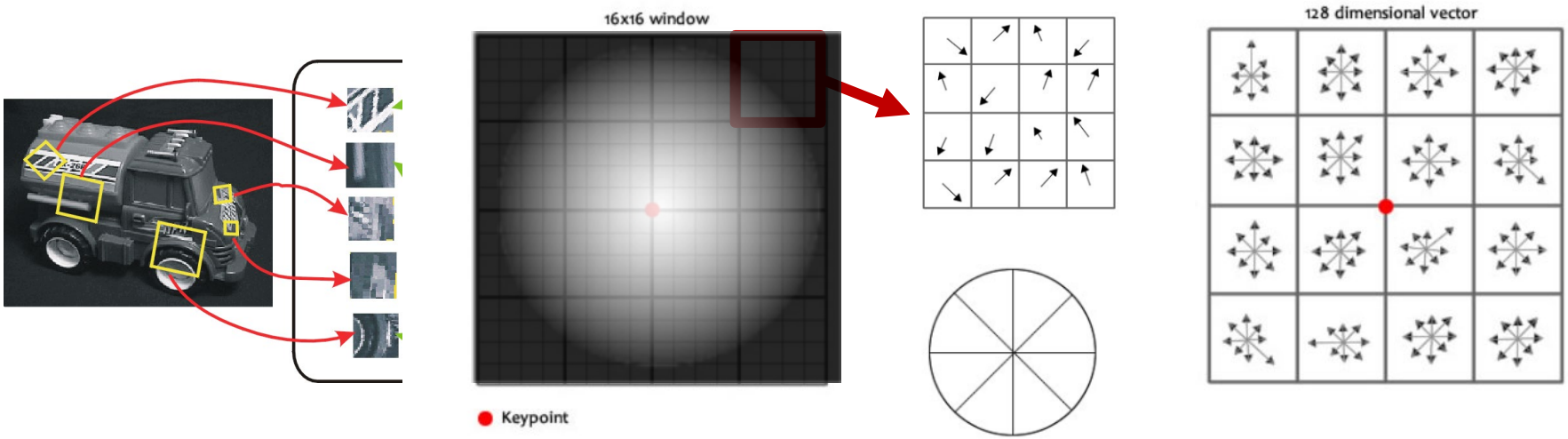
Ventana 16x16 (amarilla) en puntos de interés (rojos)
Magnitud y orientación del gradiente por píxel
Ponderado con Gaussiana
Discretiza a 8 orientaciones (*bins*)
4x4 histogramas -> 16*8=128 valores




Adaptado de James Hays y Utkarsh Sinha

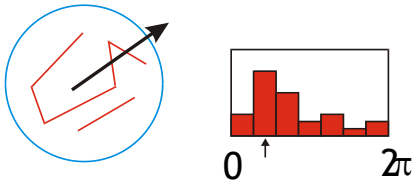
# SIFT

128 valores positivos
Normaliza para reducir efectos de contraste, ganancia
Satura gradientes > 0.2 (destellos, no lineal), y normaliza




Adaptado de James Hays y Utkarsh Sinha

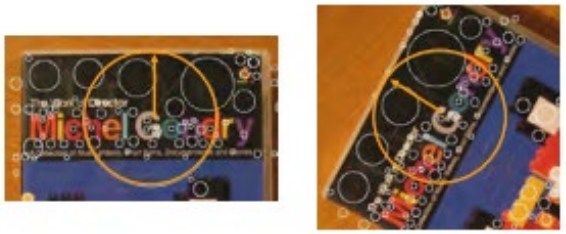
# SIFT




[Lowe, SIFT, 1999]

Parte de puntos de interés (p.e. Harris o DoG):

- Gradientes, orientación dominante
- Normaliza orientación

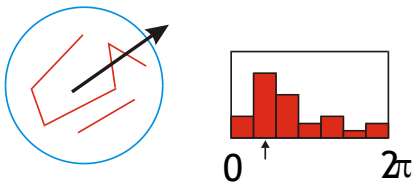


S. Nayar

```
sift = cv2.SIFT_create()
kp = sift.detect(gris,None)
imgkp=cv2.drawKeypoints(gris,kp,img)
```

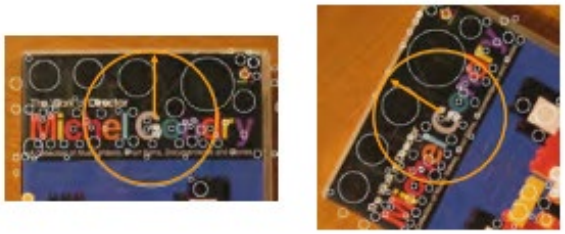
# SIFT

Parte de puntos de interés (p.e. Harris o DoG):




[Lowe, SIFT, 1999]

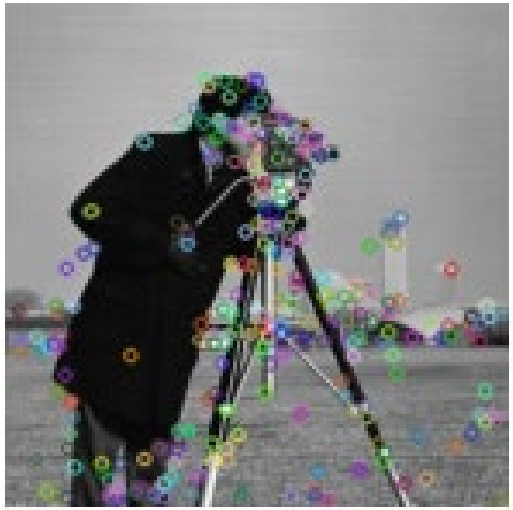
- Gradientes, orientación dominante
- Normaliza orientación



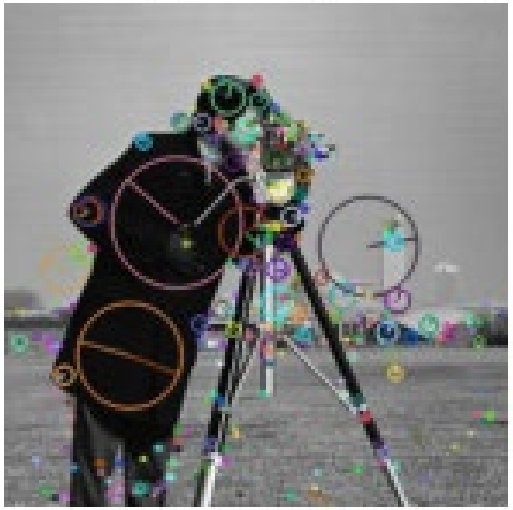
S. Nayar

Original

SIFT



SIFT (tam. y ori.)



# SIFT

Gradiente, invarianza intensidad

Orientación dominante en 16x16 para invarianza

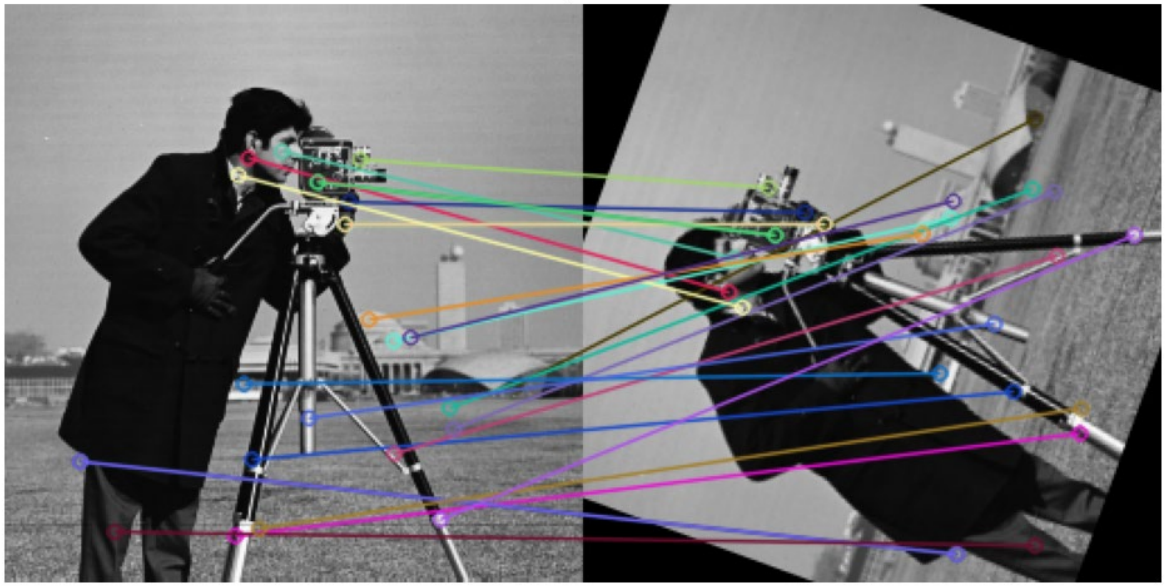
Discretiza orientaciones

Textura depende de vista, histograma para invarianza

Satura gradiente evitando brillos/destellos

Múltiples escalas

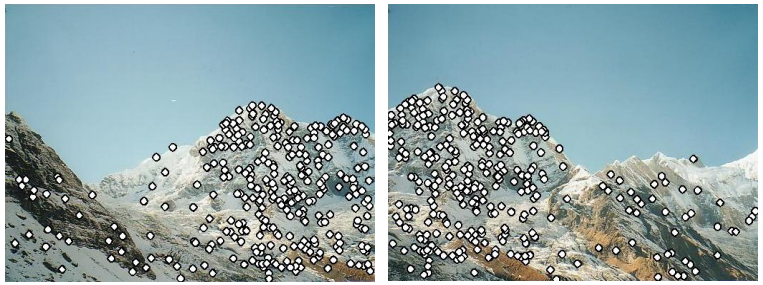
Encajes SIFT



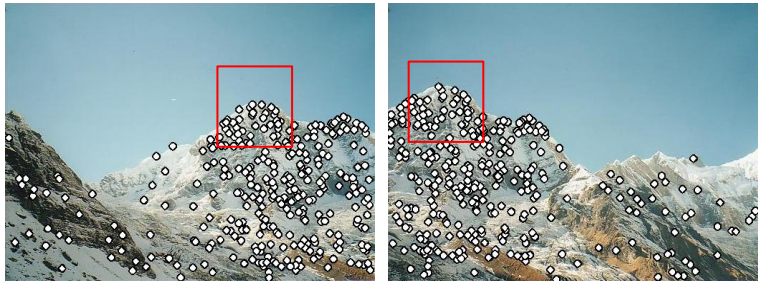
# Proceso

Etapas:

- Detección
  Localizar puntos de interés

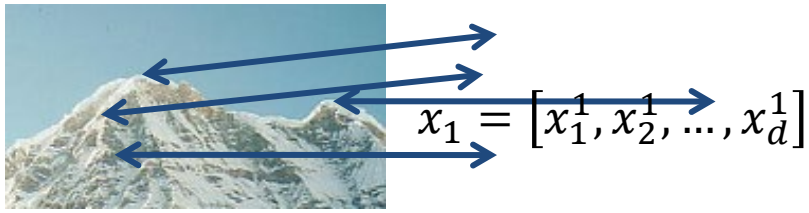


- Descripción
  Caracterizar puntos de interés



$$x_1 = [x_1^1, x_2^1, \dots, x_d^1] \qquad x_2 = [x_1^2, x_2^2, \dots, x_d^2]$$

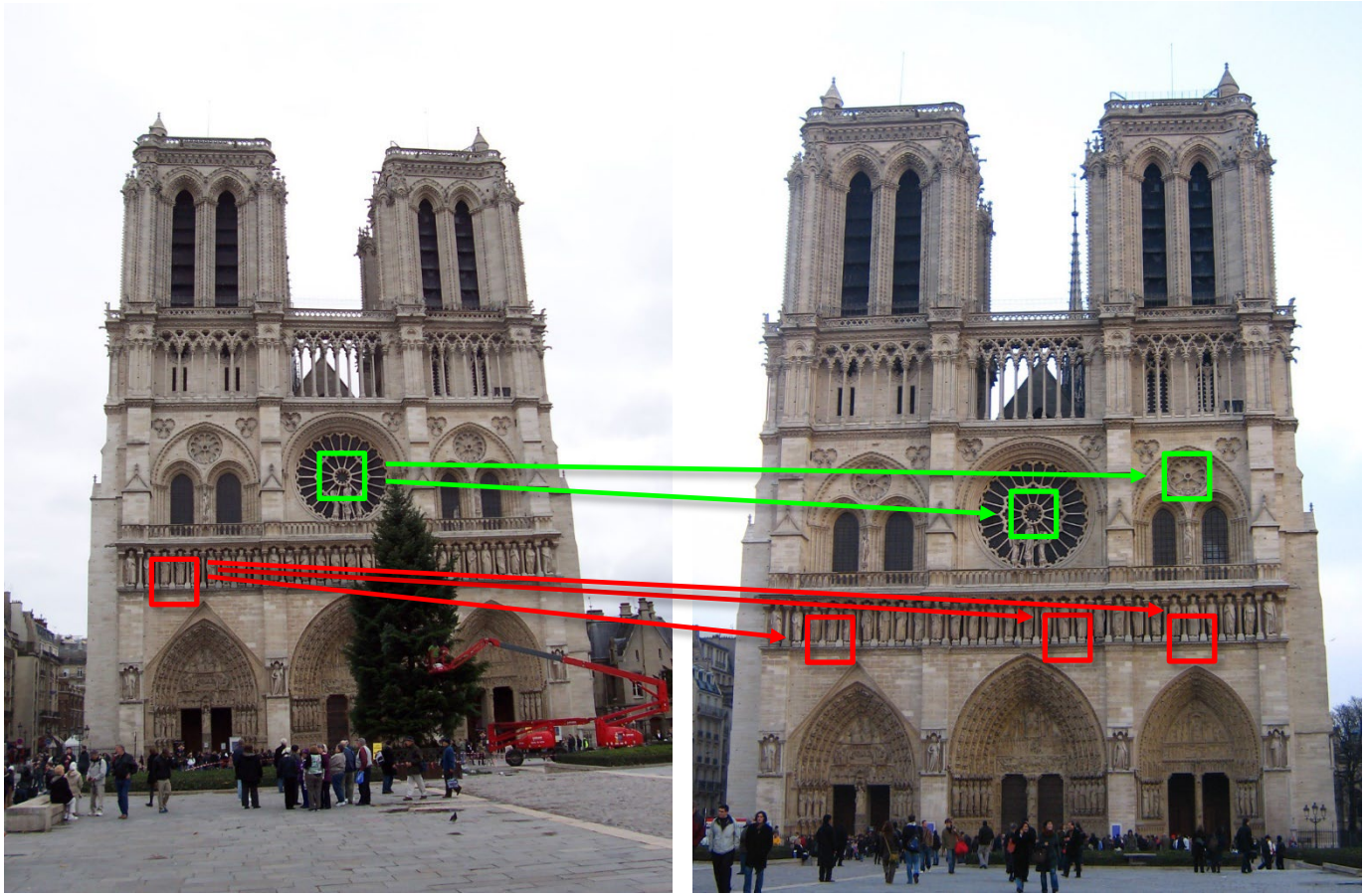
- **Encaje**
  Determinar la correspondencia




Szeliski

# Encaje

Decidir si dos puntos se corresponden



Distancia: 0.34, 0.30, 0.40

Distancia: 0.61, 1.22

Szeliski

# Encaje

Decidir si dos puntos se corresponden

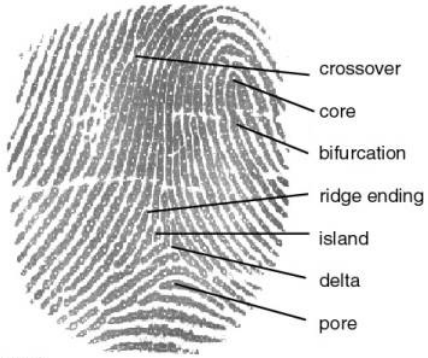
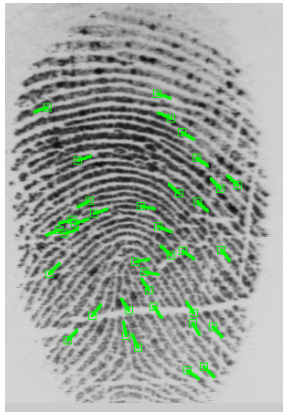



Figure 1

Calidad = 0.96
Falsas minucias=0

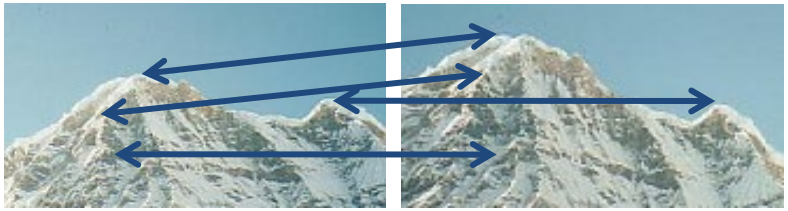


Calidad = 0.53
Falsas minucias=7

# Encaje

Dos imágenes, dos conjuntos de puntos con sus descriptores

$$x_1 = \left[x_1^1, x_2^1, \dots, x_d^1\right]$$



Szeliski

Plantillas: SSD o similar

Histogramas: Comparación

# Encaje

Descriptores SIFT, 128 valores

Distancias entre los puntos de dos imágenes

Menor distancia

Vecino más cercano (*Nearest Neighbor* NN) es $NN_1$

$$d_2(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

# Encaje

¿Todos con todos? Buf

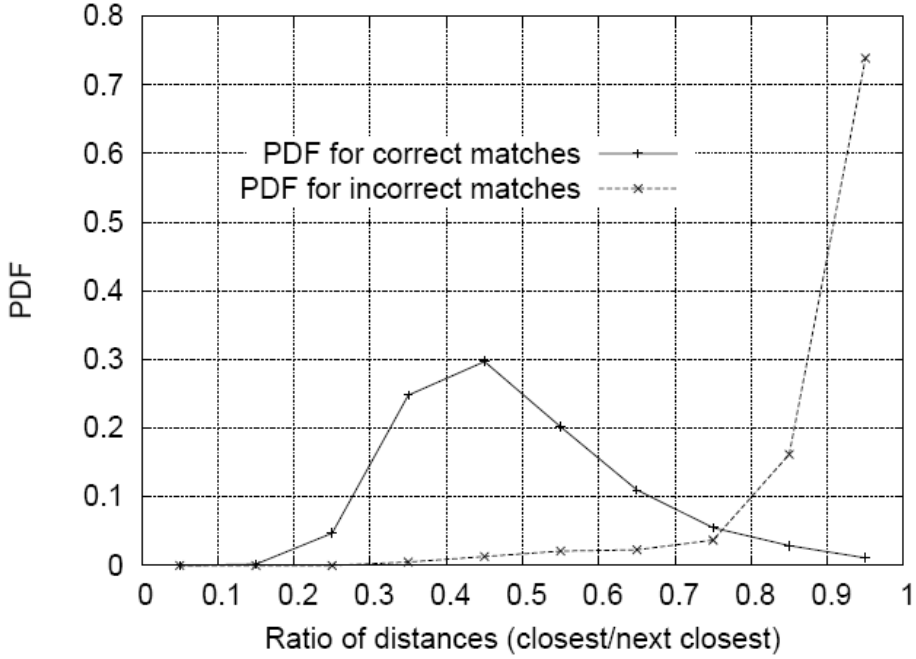
$$d_2(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Costoso
Bucles mala idea, cálculo matricial
Estructura en árbol o tabla *hash*
Vídeo, limitar búsqueda a zonas próximas

# Encaje

¿Y si el segundo más cercano tiene distancia similar?

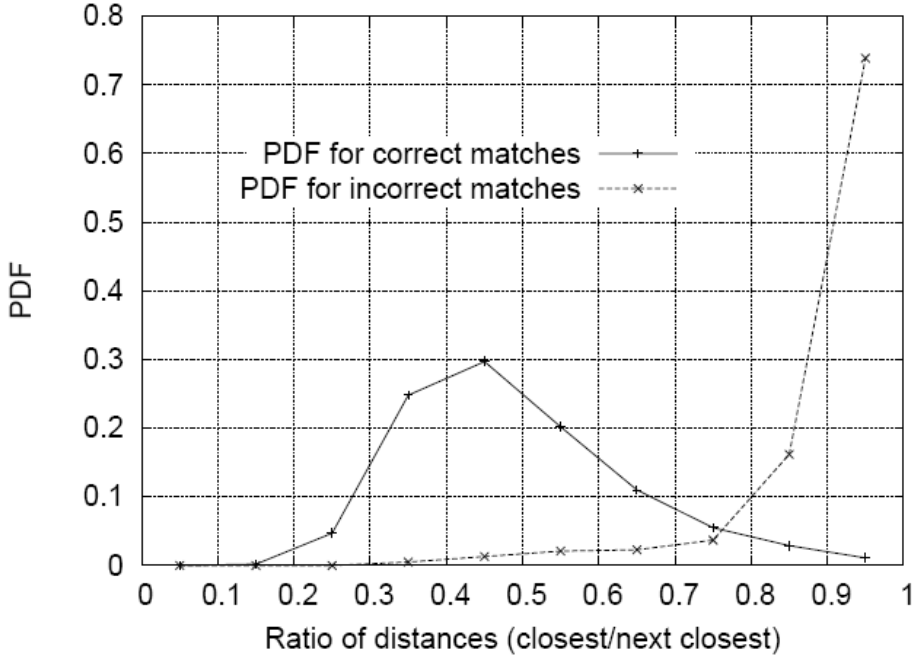
D. Lowe analizó ratio $NN_1/NN_2$ de 40000 puntos
Valores próximos a 1 *peligrosos*
Valor de confianza




D. Lowe

# Encaje

¿Tienen todos encaje?
Umbral de distancia mínima aceptable
Valor de confianza según aplicación, compromiso FP y TP
¿Qué son FP y TP?




D. Lowe

# Encaje

Evaluación

- ¿Aciertos?
- ¿Fallos?

# Encaje

## Evaluación (*scores*):

- Verdadero positivo (TP *true positive*): Encajes correctos
- Falso negativo (FN *false negative*): Encajes no detectados
- Falso positivo (FP *false positive*): Encajes incorrectos
- Verdadero negativo (TN *true negative*): No encajes bien rechazados

## Matriz de confusión

| | | Predicho | | |
|---|---|---|---|---|
| | | Encajes predichos | No encajes predichos | Total |
| Real | Encajes | TP | FN | P |
| | No encajes | FP | TN | N |
| | Total | P’ | N’ | |

# Encaje

## Matriz de confusión

| | | Predicho | | |
|---|---|---|---|---|
| | | Encajes predichos | No encajes predichos | Total |
| Real | Encajes | TP | FN | P |
| | No encajes | FP | TN | N |
| | Total | P’ | N’ | |

$$precisión\ o\ PPV = \frac{TP}{TP + FP} = \frac{TP}{P'}$$

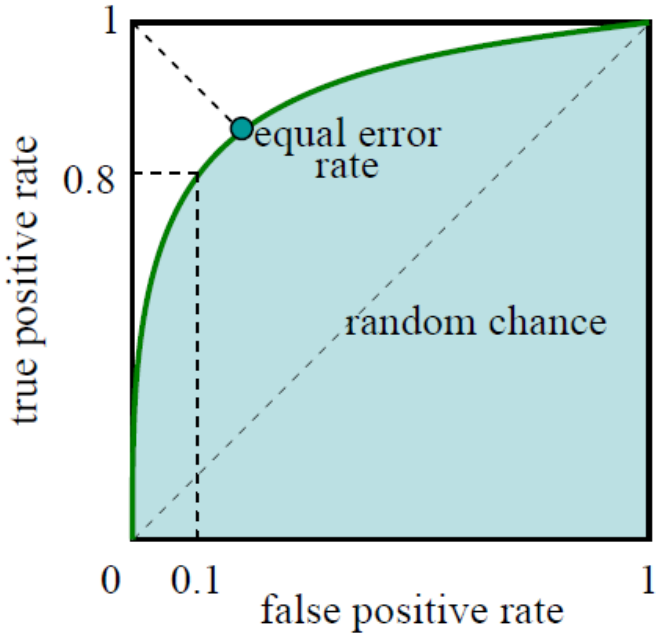
$$FPR = \frac{FP}{FP + TN} = \frac{FP}{N}$$

$$recall/sensitivity\ o\ TPR = \frac{TP}{TP + FN} = \frac{TP}{P}$$

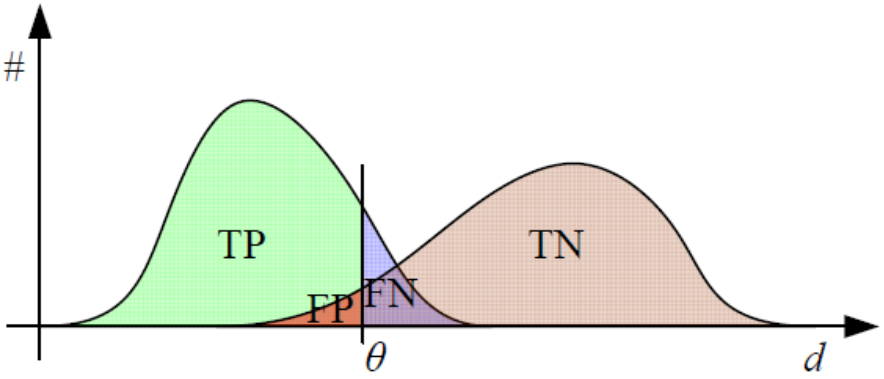
$$ACC = \frac{TP + FN}{P + N}$$

# Encaje

Dada una distancia decidir si hay encaje o no
Umbral de decisión ($\theta$)
Elección no exenta de problemas
*Receiver operating characteristic* (ROC) enfrenta TPR a FPR
Comparativa de modelos con curva ROC




(a)




(b)

Szeliski

# Encaje

**Distancia euclídea**

No la única

Elección dependiente del problema

$$d_2(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

# Encaje

**Distancia euclídea**
No la única
Elección dependiente del problema

$$d_2(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Distancias entre vectores de características

*Dos muestras a y b con n componentes*

- Distancia de **Minkowski:**

$$d(a,b) = \left(\sum_{i=1}^{n}|a_i - b_i|^r\right)^{\frac{1}{2}}, r \geq 1$$

- Distancia de **Mahalanobis:**
  - Interviene la correlación de las variables
  - No afecta la escala de la medida
  - Σ matriz de covarianza

$$d_M(a,b) = \sqrt{(a-b)^T \Sigma^{-1}(a-b)}$$

# Encaje

Concreciones de Minkowski:

- Manhattan o city block (r=1)
- Euclídea o L2 (r=2)
- Chebyshev o tablero de ajedrez (r -> ∞)

$$d_1(a,b) = \sum_{i=1}^{n} |a_i - b_i|$$

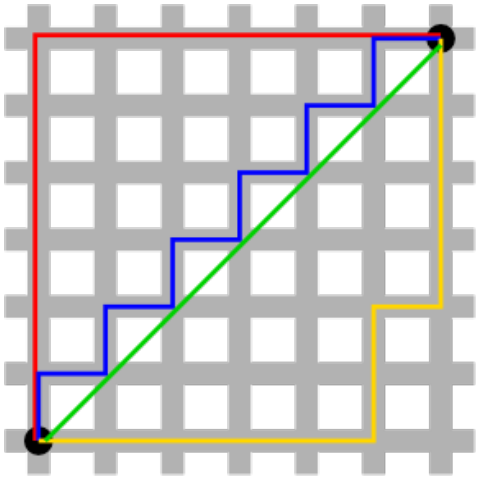
$$d_2(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

$$d_\infty(a,b) = máx_{i=1..n} |a_i - b_i|$$

Manhattan = 12

Euclídea ≈ 8.5

Chebyshev = 6



*d = 0* para el mejor encaje

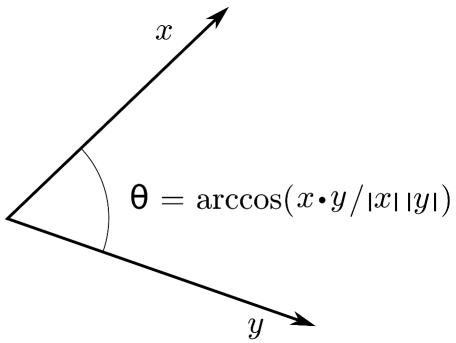
*sklearn.neighbors.DistanceMetric*

# Encaje

## Similitud coseno

Coseno del ángulo formado entre dos vectores

$$\cos\theta = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|}$$




Rango [ -1, 1 ]

Resulta 1 cuando son idénticos

No es métrica/distancia, no cumple la desigualdad triangular

# Encaje

## **Comparación de histogramas** de *n bins*, en OpenCV:

- Correlación (1 idénticos)

$$d(h_1, h_2) = \frac{\sum_i \left(h_1(i) - \overline{h}_1\right)\left(h_2(i) - \overline{h}_2\right)}{\sqrt{\sum_i \left(h_1(i) - \overline{h}_1\right)^2 \sum_i \left(h_2(i) - \overline{h}_2\right)^2}} \quad \text{donde} \quad \overline{h}_k = \frac{1}{n}\sum_j h_k(j)$$

- Chi-square (0 idénticos)

$$d(h_1, h_2) = \sum_i \frac{\left(h_1(i) - h_2(i)\right)^2}{h_1(i)}$$

- Intersección (mayores más similares)

$$d(h_1, h_2) = \sum_i min(h_1, h_2)$$

- Bhattacharyya (0 idénticos)

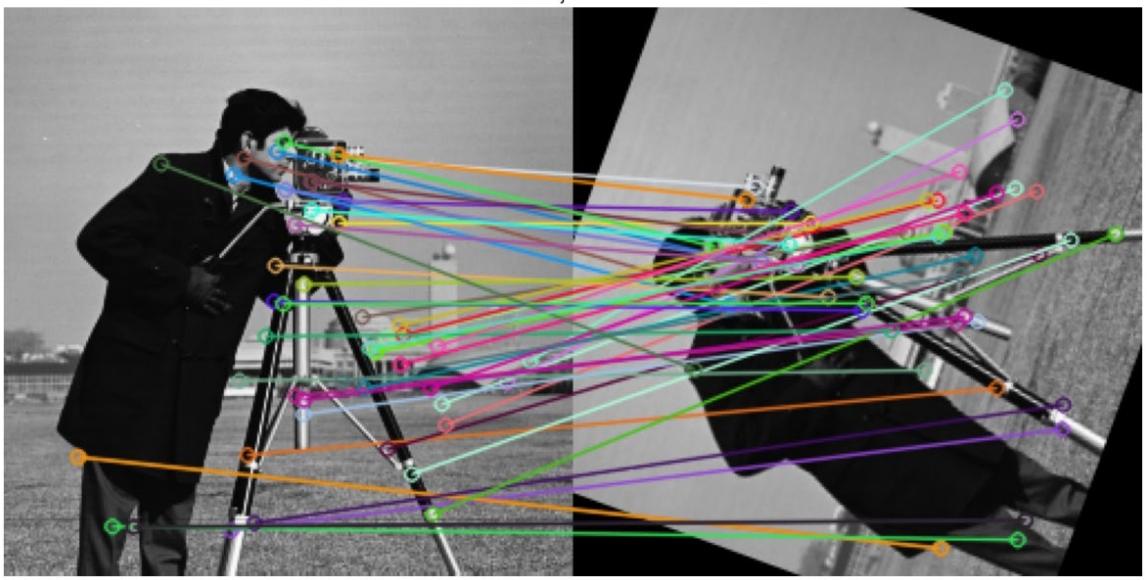
$$d(h_1, h_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{h}_2 \overline{h}_2 n^2}} \sum_i \sqrt{h_1(i) \cdot h_2(i)}}$$

# Encaje

## Encaje puntos SIFT en OpenCV

```
sift = cv2.SIFT_create()
kp1,de1 = sift.detectAndCompute(gris1,None)
kp2,de2 = sift.detectAndCompute(gris2,None)
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches = bf.match(de1, de2)
```

Encajes SIFT



# Encaje

## Encaje puntos SIFT en OpenCV

```
sift = cv2.SIFT_create()
kp1,de1 = sift.detectAndCompute(gris1,None)
kp2,de2 = sift.detectAndCompute(gris2,None)
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches = bf.match(de1, de2)
# Ordenar orrespondencias por distancia
matches = sorted(matches, key=lambda x: x.distance)
img_matches = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], None,
matchColor=(0, 0, 0),flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

Encajes SIFT

# Encaje

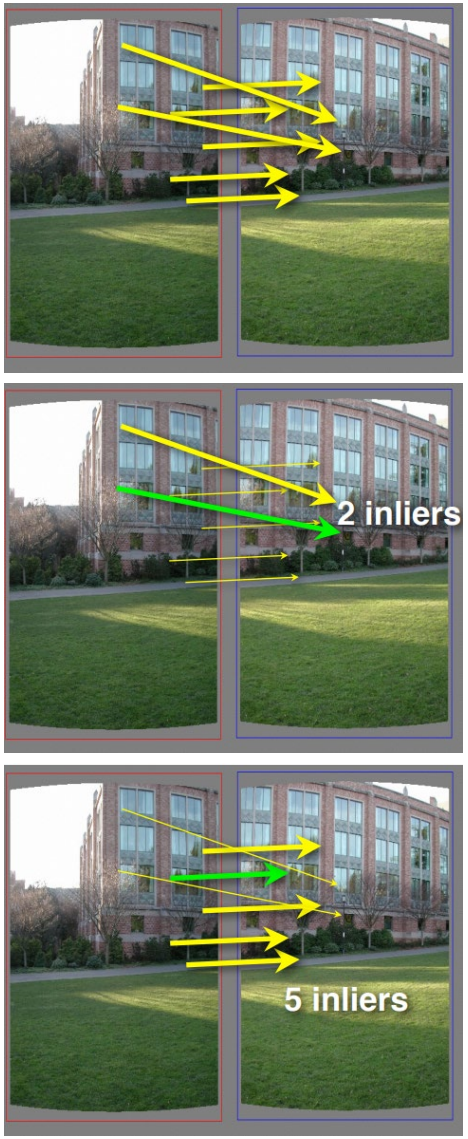
Encajes no coherentes, *outliers*

Método RANSAC (*random sample consensus*):

- Escoger par(es) aleatoriamente
- Contar *inliers*
- Repetir el proceso

Repeticiones según número muestras

Escoge la de mayor número de *inliers*




I. Gkioulekas

# Encaje

Eliminar encajes erróneos (*outliers*) con [cv.findHomography]

```
# Extrae los puntos de encaje
pts1 = np.float32([kp1[m.queryIdx].pt for m in matches]).reshape(-1, 1, 2)
pts2 = np.float32([kp2[m.trainIdx].pt for m in matches]).reshape(-1, 1, 2)
# RANSAC para filtrar outliers
H, mask = cv2.findHomography(pts1, pts2, cv2.RANSAC, 5.0)
matches_f = mask.ravel().tolist()
# Filtra correspondencias con RANSAC
matchesf = [matches[i] for i in range(len(matches)) if matches_mask[i]]
```

Encajes SIFT

# Encaje

## Encaje puntos ORB (2011) en OpenCV

```
orb = cv2.ORB_create()
kp1,de1 = orb.detectAndCompute(gris1,None)
kp2,de2 = orb.detectAndCompute(gris2,None)
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(de1, de2)
# Ordenar orrespondencias por distancia
matches = sorted(matches, key=lambda x: x.distance)
img_matches = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], None,
matchColor=(0, 0, 0),flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

Encajes ORB

# Encaje

Encaje puntos ORB en OpenCV
Tras eliminar *outliers* con RANSAC
OpenCV ofrece SIFT, SURF y ORB
ORB (2011) sensiblemente más rápido y sensible

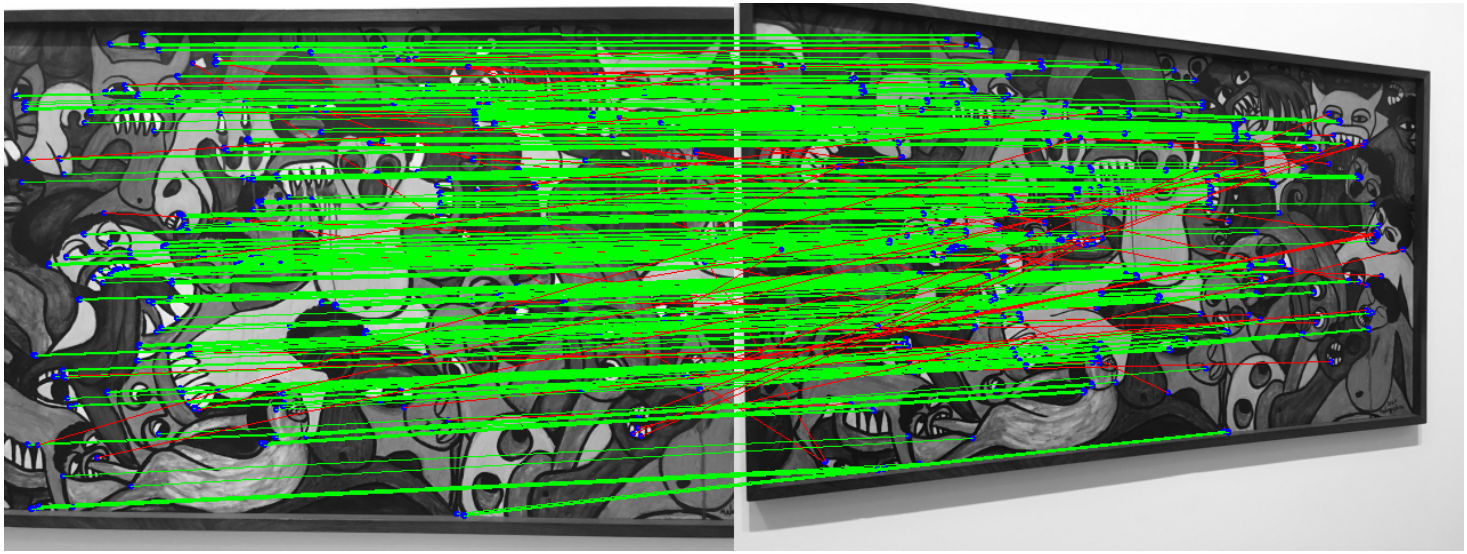
Encajes ORB

# ¿Y en tiempos de DL?

SuperPoint (2018)
Detección y descripción similar a SIFT
Basado en VGG
70 fps

# ¿Y en tiempos de DL?

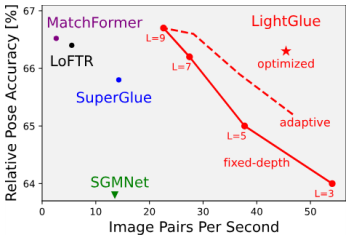
SiLK - Simple Learned Keypoints (2023)
Dos GPUs para entreno
30 fps

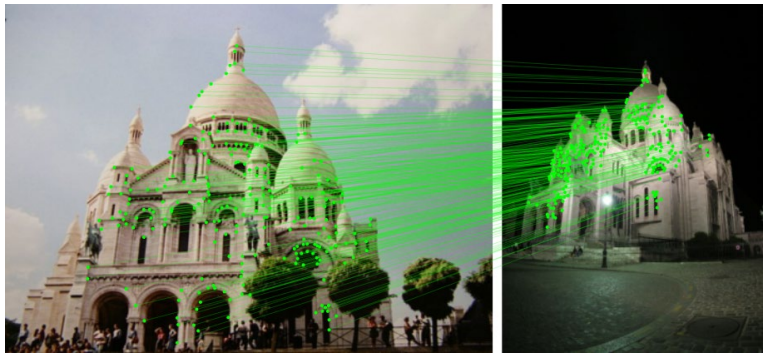


# ¿Y en tiempos de DL?

Encaje

LightGlue (2023)
Puntos



GlueStick (2023)
Puntos y líneas

# ¿Qué es un borde/contorno?

# ¿Qué es un borde/contorno?

**Borde**: frontera entre zonas
**Región**: zona homogénea delimitada por bordes
Dualidad borde/región

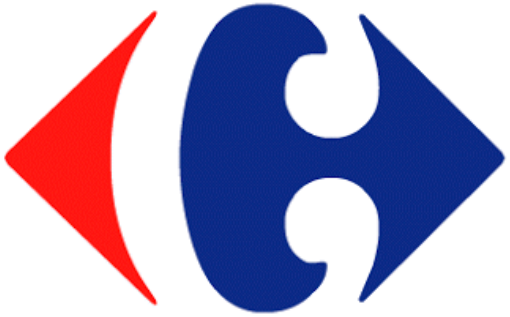
Importancia en la percepción

Flautista, Picasso

Jean Cocteau

# ¿Qué es un borde/contorno?

## Bordes ilusorios



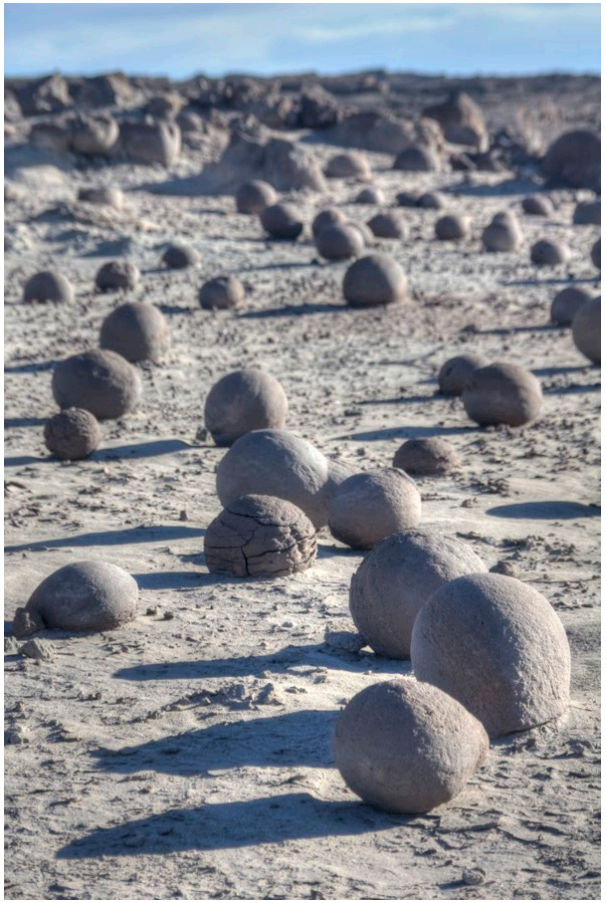
# ¿Qué es un borde/contorno?

Importancia en análisis automático de imágenes

# ¿Qué es un borde/contorno?

Discontinuidades en intensidad:

- Profundidad: Cambio brusco de distancia
- Normal superficie: Orientación de la superficie
- Iluminación: Sombras, cambio de iluminación
- Reflexión: Propiedades de la superficie, marcas, textura

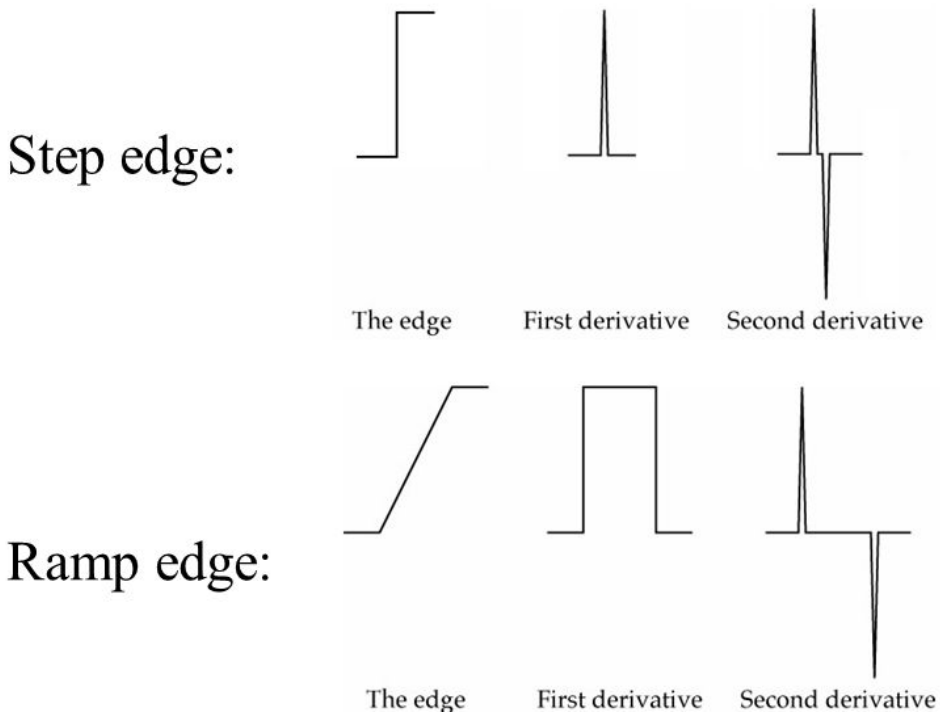


Realidad ruidosa, no salto limpio

# ¿Cómo se detecta?

Borde ideal (sin ruido), escalón o rampa:

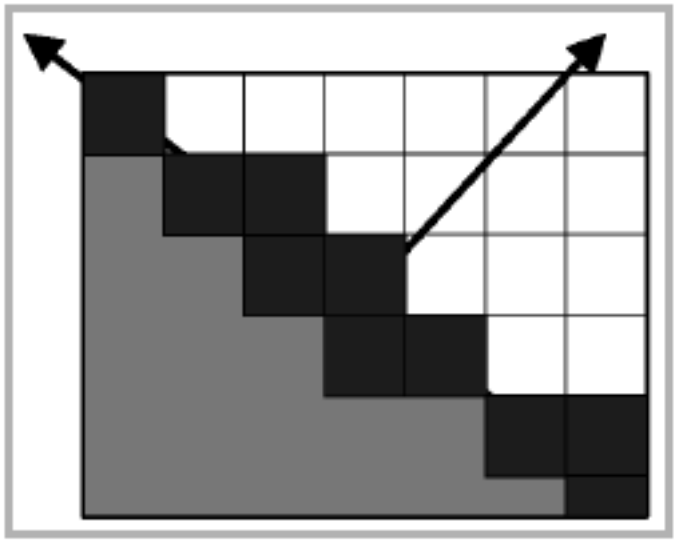
- Primera derivada (gradiente) valor alto -> pico
- Segunda derivada -> cruce/paso por cero



# ¿Cómo se detecta?

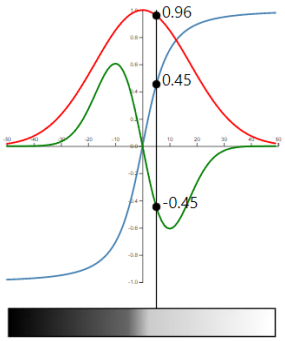
**Propiedades** de un borde:

- **Normal**: Dirección de máximo cambio de intensidad
- **Dirección**: Dirección perpendicular a la normal
- **Posición**: En imagen
- **Fuerza**: Velocidad cambio de intensidad



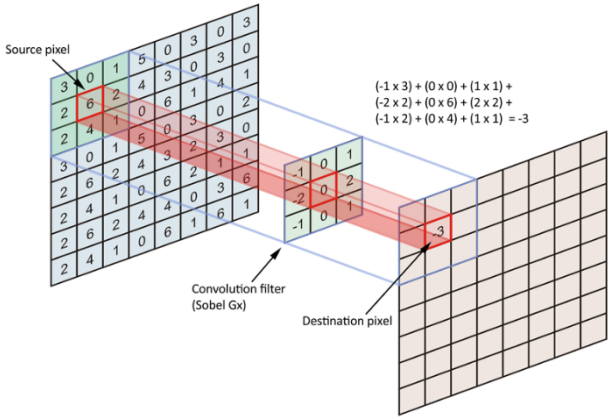
P. Fua

# ¿Cómo se detecta?




Operadores detección de bordes:

- Primera derivada: gradiente
- Segunda derivada: laplaciana
- Discretizar *(kernel)* y convolucionar: Prewitt, Sobel, …
- Aprendizaje automático

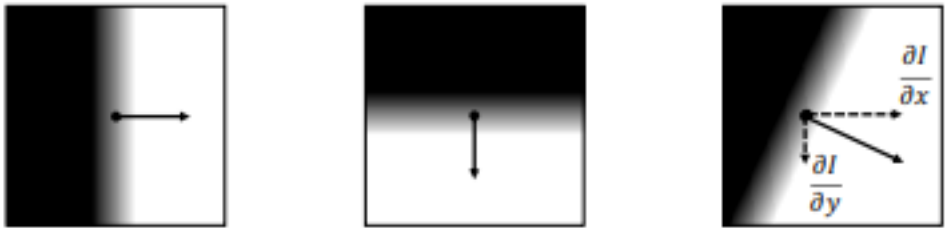



$$contornos\ imagen = contornos\ verticales + contornos\ horizontales$$

# Gradiente

Gradiente
Mide cambio local: diferencia de intensidad
Describe magnitud y dirección del cambio
Componentes *x* e *y*, magnitud y orientación

$$J(x) = \nabla f(x) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)(x)$$



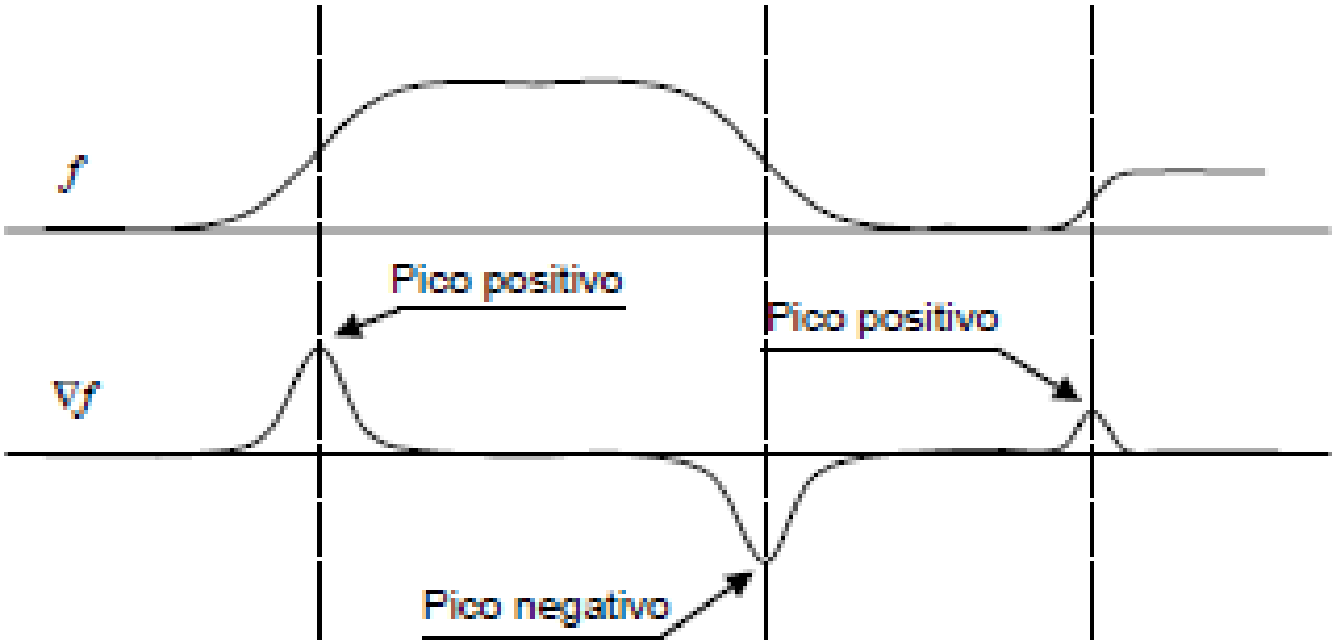

$\nabla I = \left[\frac{\partial I}{\partial x}, 0\right]$ $\nabla I = \left[0, \frac{\partial I}{\partial y}\right]$ $\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$

S. Nayar

# Gradiente

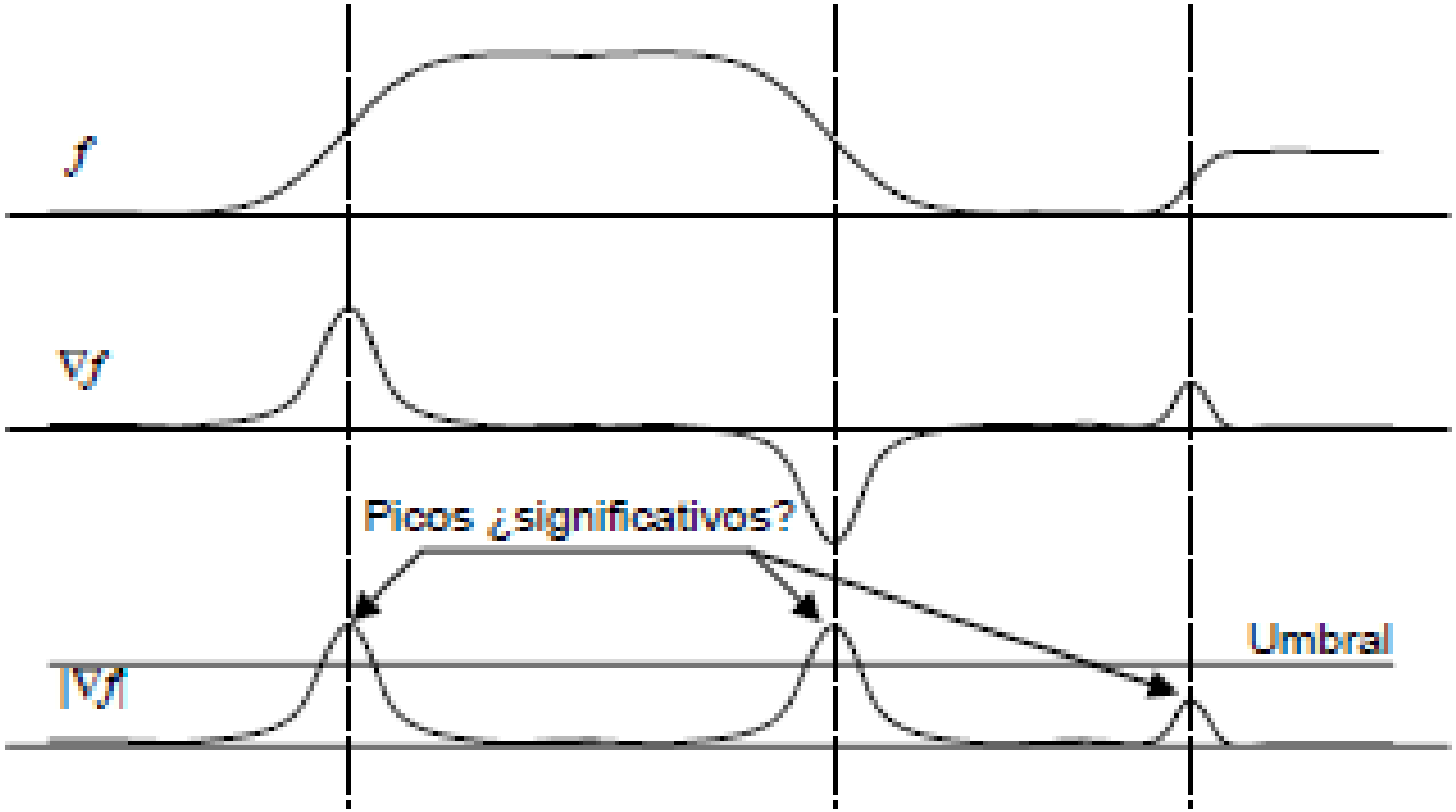
Picos de magnitud
Diferencias altas, evidencia de borde

$$\nabla f(x) = \frac{\partial f}{\partial x}$$



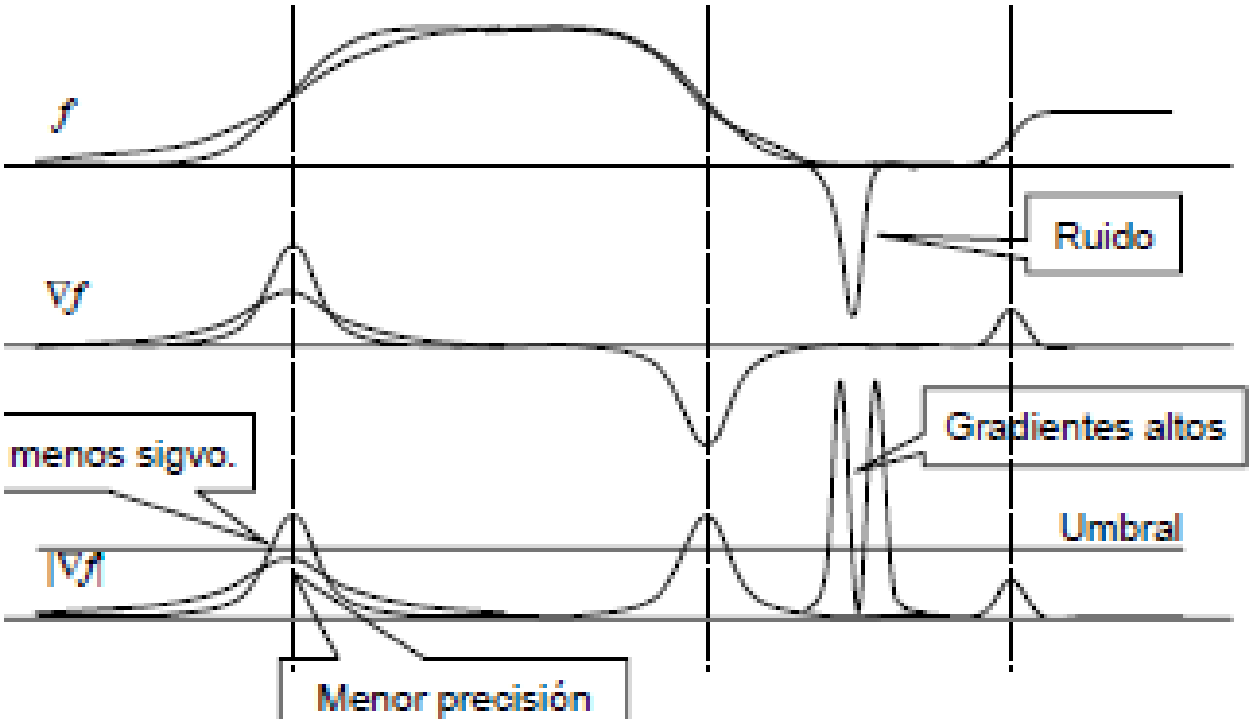
# Gradiente

Picos de magnitud
Diferencias altas, evidencia de borde
Umbral para decidir



# Gradiente

El gradiente responde al ruido (falsos picos)
Suavizado, elimina altas frecuencias
Convolución con filtro paso bajo -> Gaussiana
Perjudica localización del borde
Mejor enfoque -> Canny

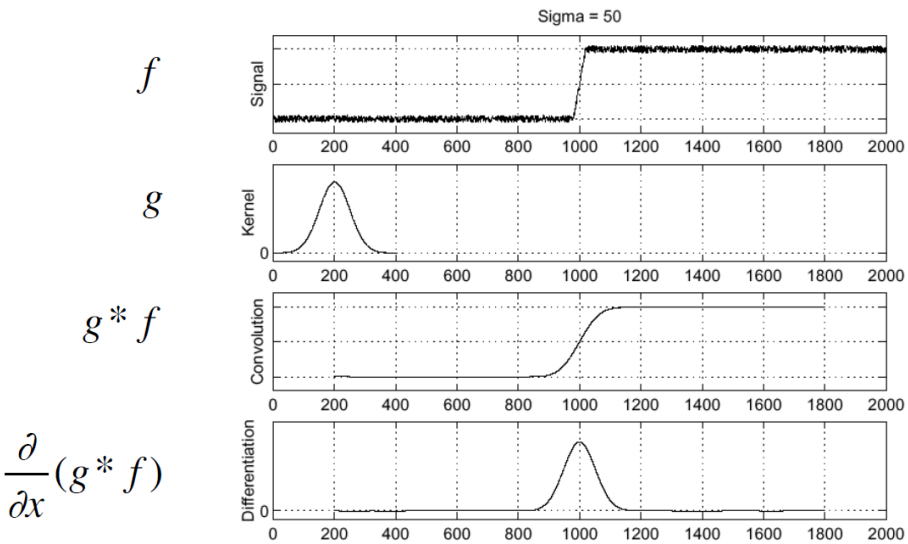


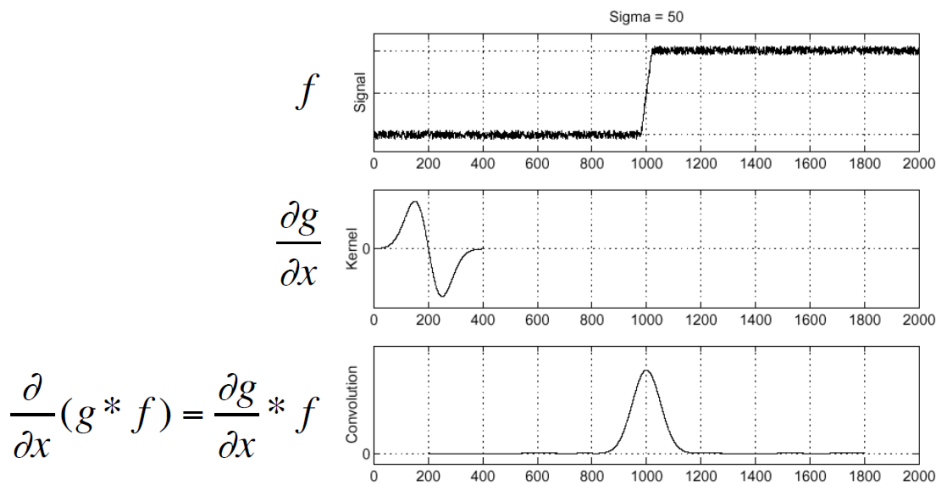
# Gradiente

Convolución con filtro paso bajo -> Gaussiana
Gaussiana previa o simultánea

$$J_\sigma(x) = \nabla[G_\sigma * f(x)] = [\nabla G_\sigma](x) * f(x)$$

Gradiente de Gaussiana precalculado $\frac{\partial G}{\partial x}$




$$\frac{\partial}{\partial x}(g * f) = \frac{\partial g}{\partial x} * f$$

P. Fua

# Gradiente

Aproximar gradiente con convolución con *kernel*
Horizontal y vertical

*Kernel* de menor dimensión:

- más sensible al ruido
- mejor localización
- peor detección

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Prewitt 3x3

*Kernel* de mayor dimensión:

- menos sensible al ruido
- peor localización
- mejor detección

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

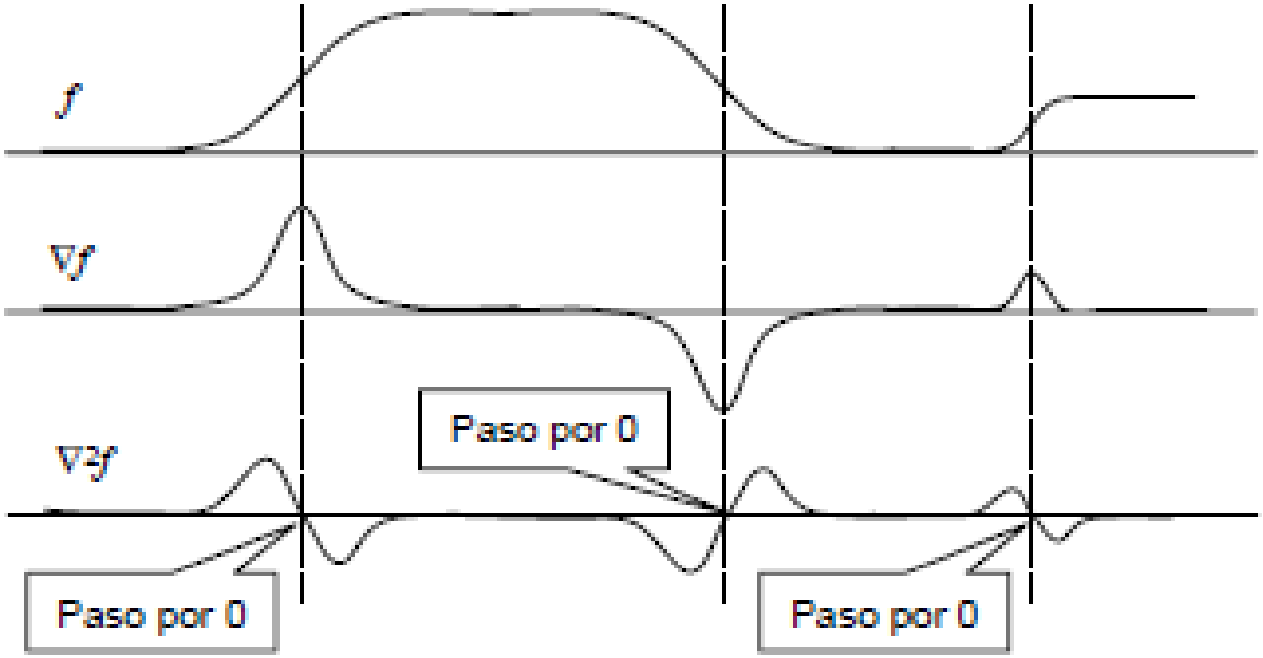
| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Sobel 3x3

# Laplaciana

Laplaciana
Mide cambios del gradiente
Segunda derivada de *f*
Pasos por cero (picos primera derivada)
Realce de bordes

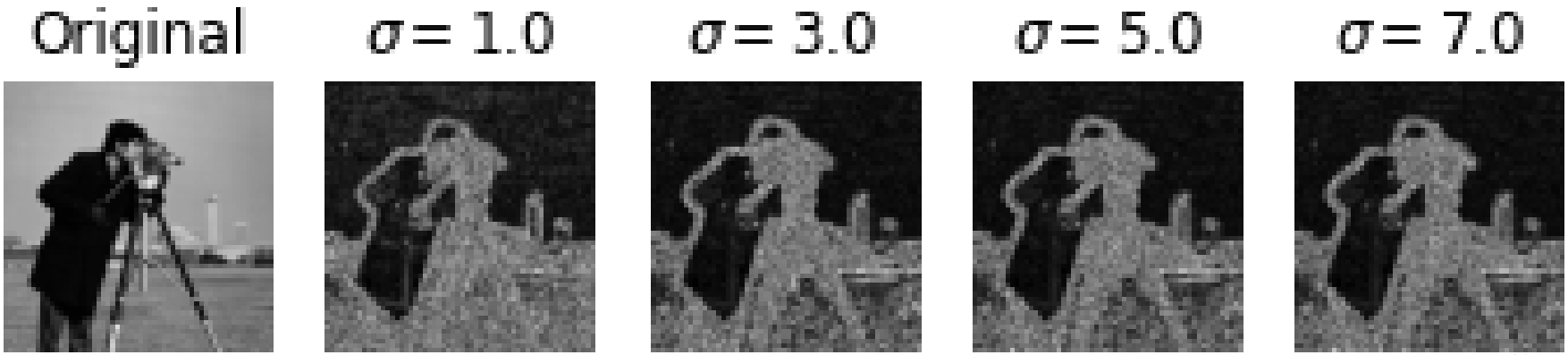
$$\nabla^2 f(x) = \frac{\partial^2 f}{\partial x^2}$$



# Laplaciana

Sensible al ruido, realza bordes
Suavizado, elimina altas frecuencias, atenúa pendientes
Convolución con filtro paso bajo -> Gaussiana
Comportamiento de bordes a distintas escalas
Solución: filtrar a distintas escalas, distintas σ

Gaussiana $\sigma\downarrow$, + sensible ruido, + bordes falsos, + precisa
Gaussiana $\sigma\uparrow$: + bordes perdidos, - precisa



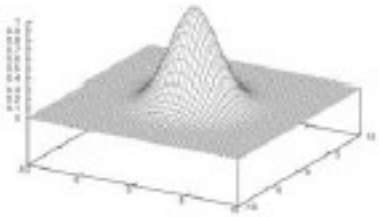
# Laplaciana

Laplaciana de Gaussiana (LoG, 1980)
Suaviza y realza
Otras denominaciones:

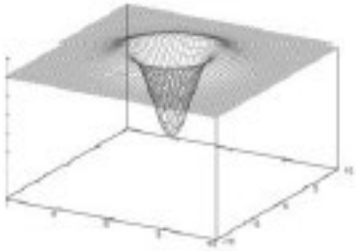
- Operador de Marr-Hildreth
- Sombrero mejicano

$$\nabla^2(f * g) = f * \nabla^2 g$$

$$f * \nabla^2 g = f * \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial y^2} g\right) = f * \frac{\partial^2}{\partial x^2} g + f * \frac{\partial^2}{\partial y^2} g$$



Gaussiana 2D



LoG 2D

# Laplaciana

Aproximar Laplaciana con convolución con *kernel*

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

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

Basta un *kernel*, es simétrico

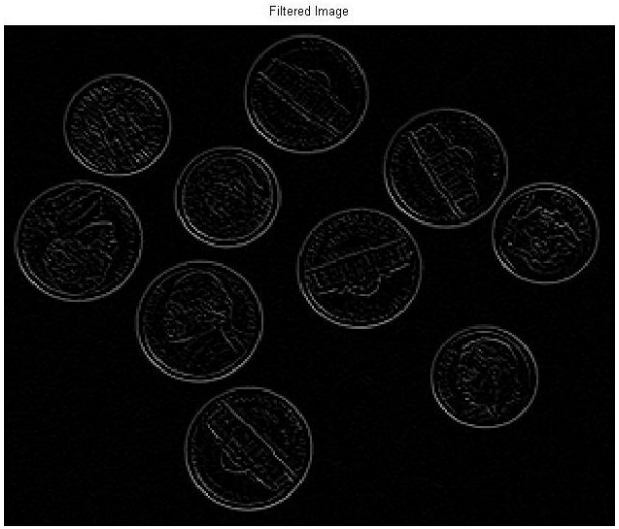
*Kernel* con Gaussiana σ=1.4

$$LoG = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |
| 1 | 2 | 4 | 5 | 5 | 5 | 4 | 2 | 1 |
| 1 | 4 | 5 | 3 | 0 | 3 | 5 | 4 | 1 |
| 2 | 5 | 3 | -12 | -24 | -12 | 3 | 5 | 2 |
| 2 | 5 | 0 | -24 | -40 | -24 | 0 | 5 | 2 |
| 2 | 5 | 3 | -12 | -24 | -12 | 3 | 5 | 2 |
| 1 | 4 | 5 | 3 | 0 | 3 | 5 | 4 | 1 |
| 1 | 2 | 4 | 5 | 5 | 5 | 4 | 2 | 1 |
| 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |

# Laplaciana

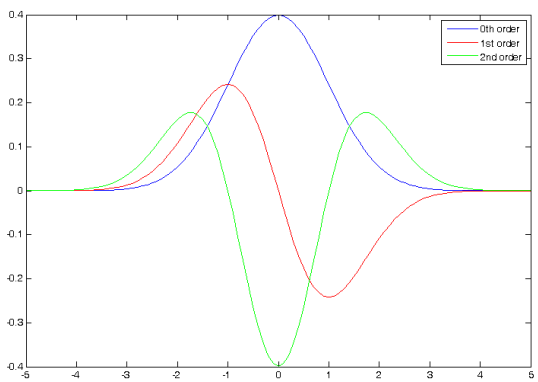
Combinar imagen original con Laplaciana
Realza bordes

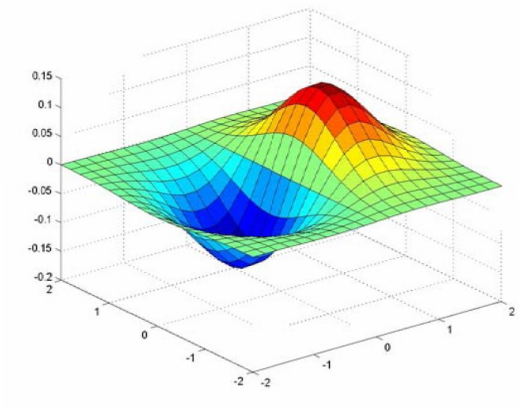


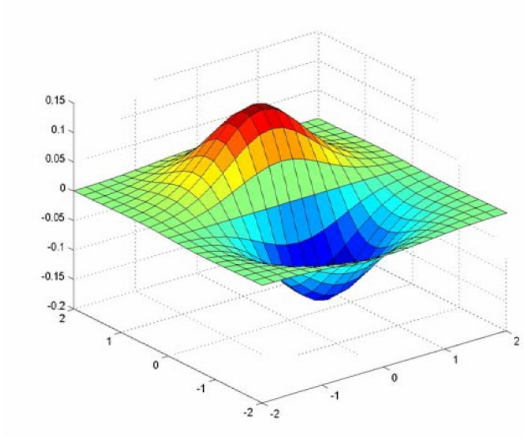
# Canny

Detector de bordes más conocido (>45k citas)

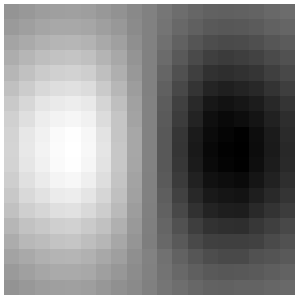
La primera derivada de la Gaussiana aproxima el operador que optimiza relación señal-ruido (SNR) y posición



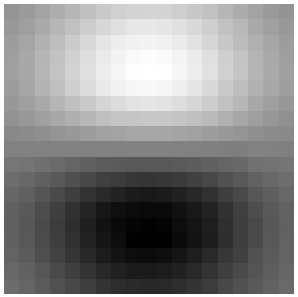





*x*



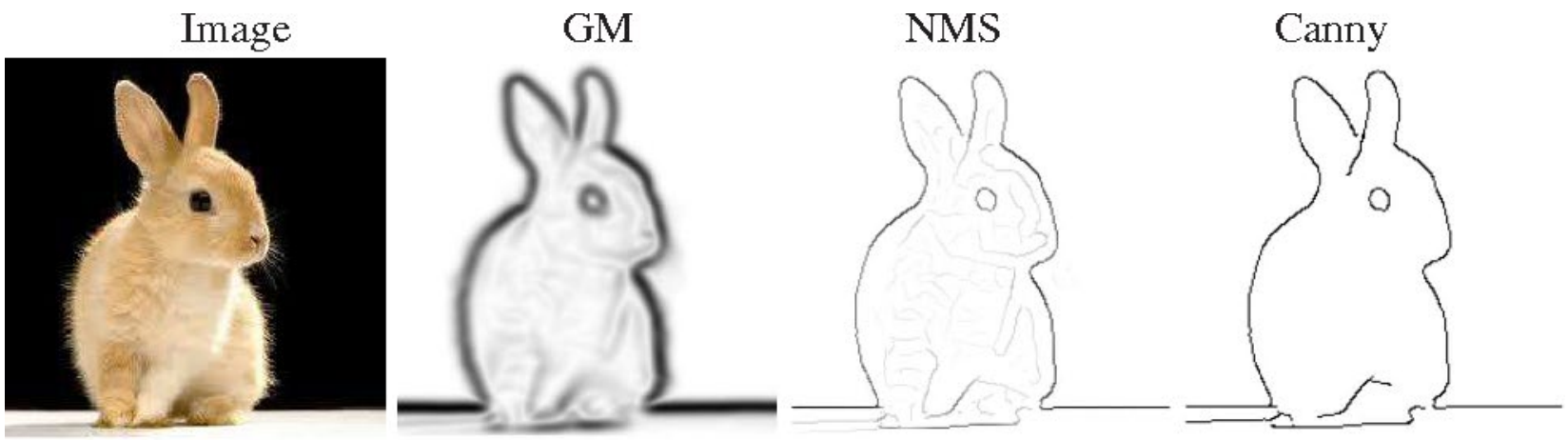
*y*



J. Canny, *A Computational Approach To Edge Detection*, PAMI, 1986.

# Canny

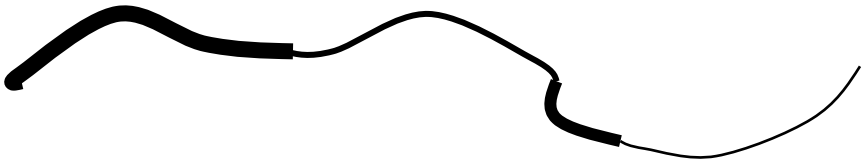
1. Gradiente (Sobel) suavizando con Gaussiana
2. Obtiene magnitud y orientación
3. Determina picos significativos
   - Elimina no máximos (NMS adelgaza bordes a un píxel)
   - Histéresis de umbral (doble umbral T1<T2)
     - Picos mayores que T2 bordes
     - Picos menores que T1 no bordes
     - Intermedios, borde si está conectado a un borde



# Canny

- Histéresis de umbral (doble umbral T1<T2)
  - Picos mayores que T2 bordes
  - Picos menores que T1 no bordes
  - Intermedios, borde si está conectado a un borde

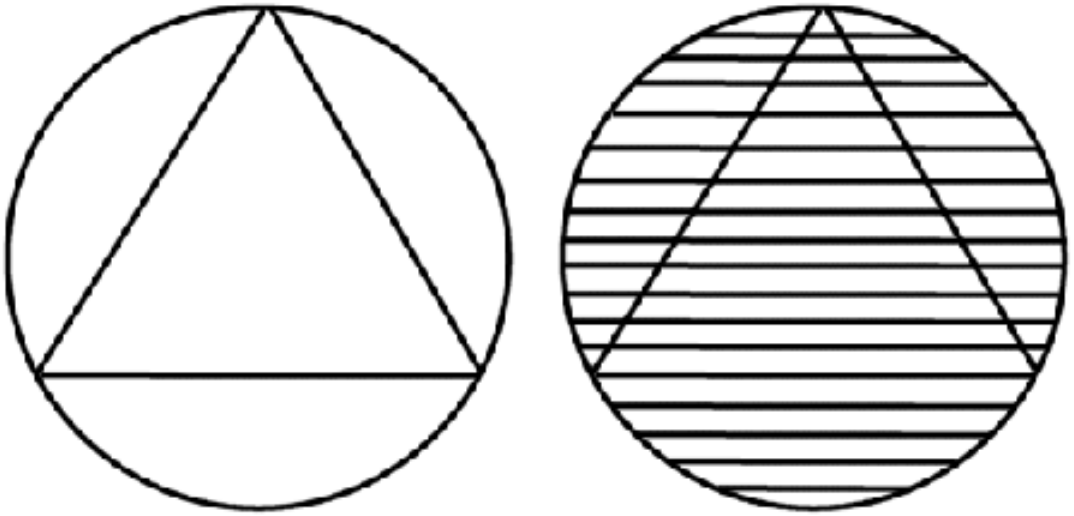
```
T1 = 75
T2 = 175
#Canny
edge = cv2.Canny(gris, T1, T2,5)
```



S. Seitz

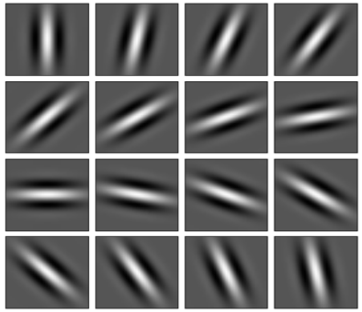
# Canny

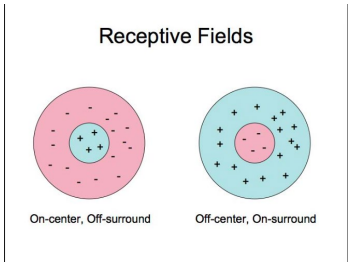
Con supresión

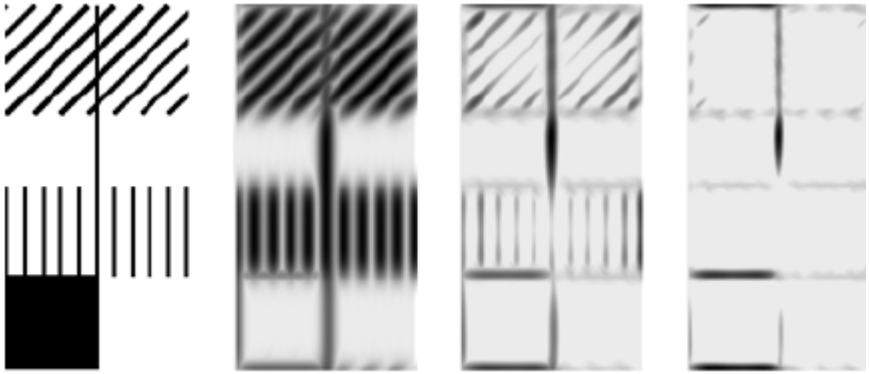


# Canny

Con supresión
Filtros de Gabor
Campos receptivos, e inhibición





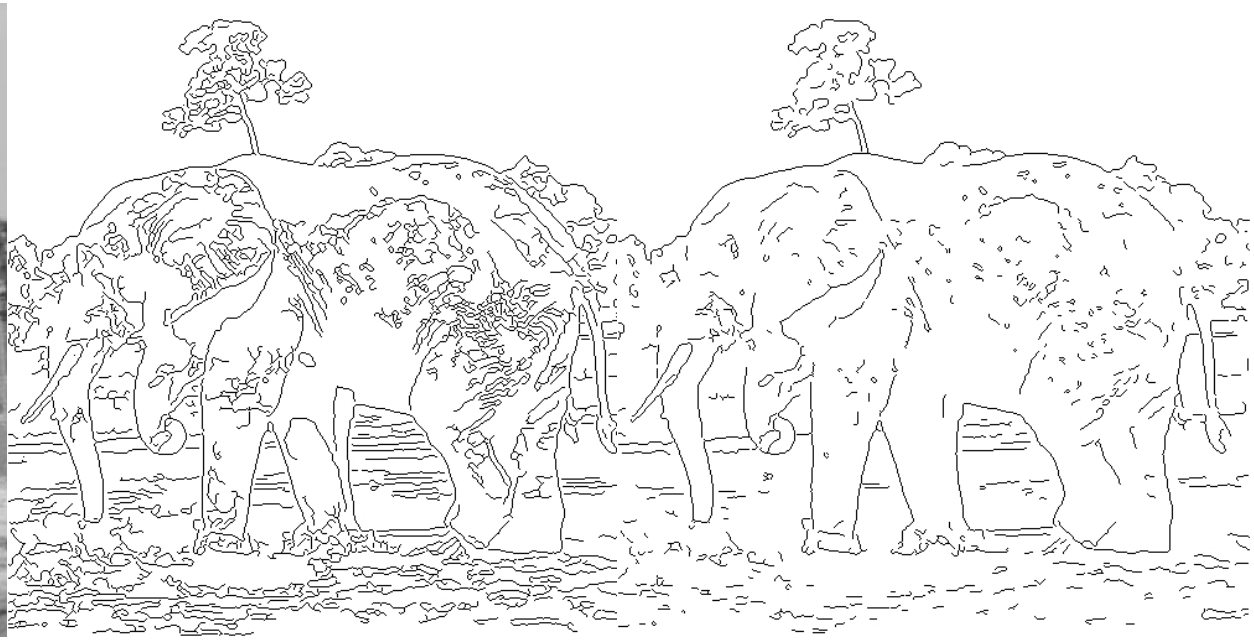



(a) (b) (c) (d)

# Canny

Con supresión
Filtros de Gabor
Campos receptivos, e inhibición

Original



Canny

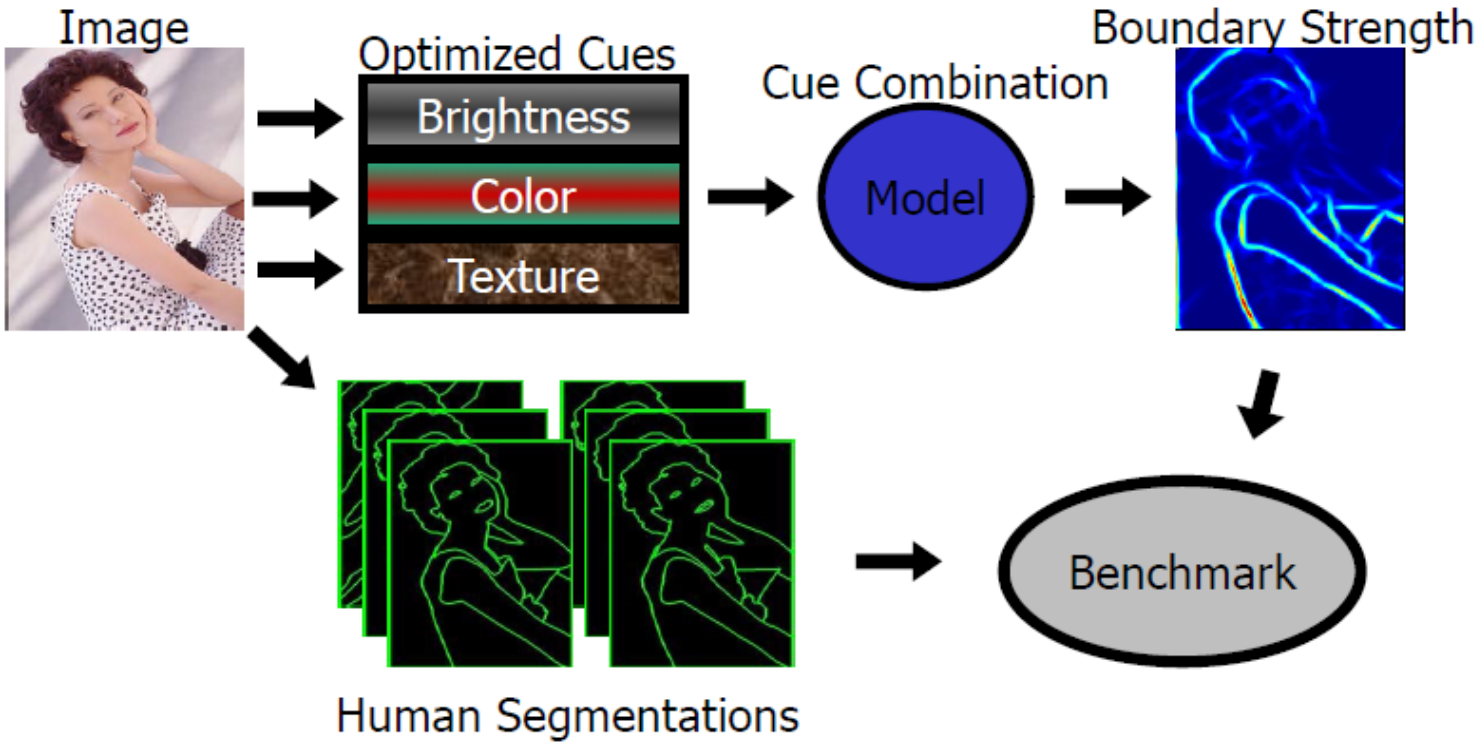
Canny y supresión

G. Papari, P. Campisi, N. Petkov, A. Neri. A biologically motivated multiresolution approach to contour detection. JASP 2006

# Aprendizaje automático

Datos anotados

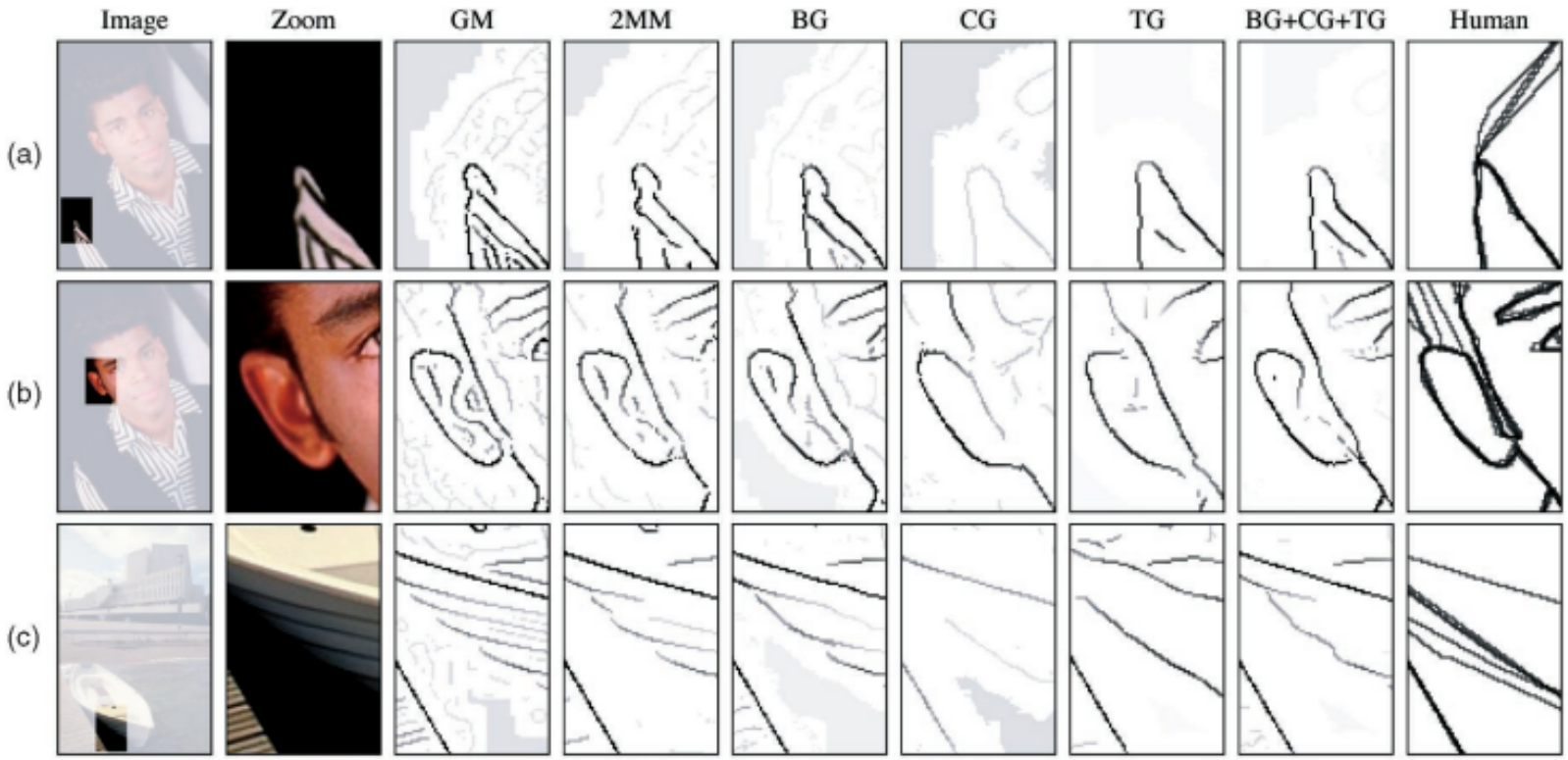
Aprender probabilidad en base a diversas propiedades




P. Fua

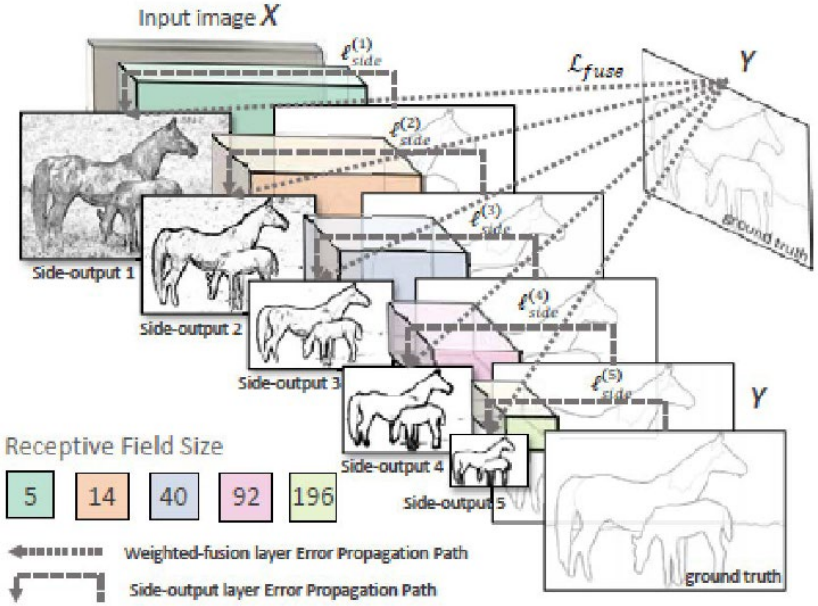
# Aprendizaje automático

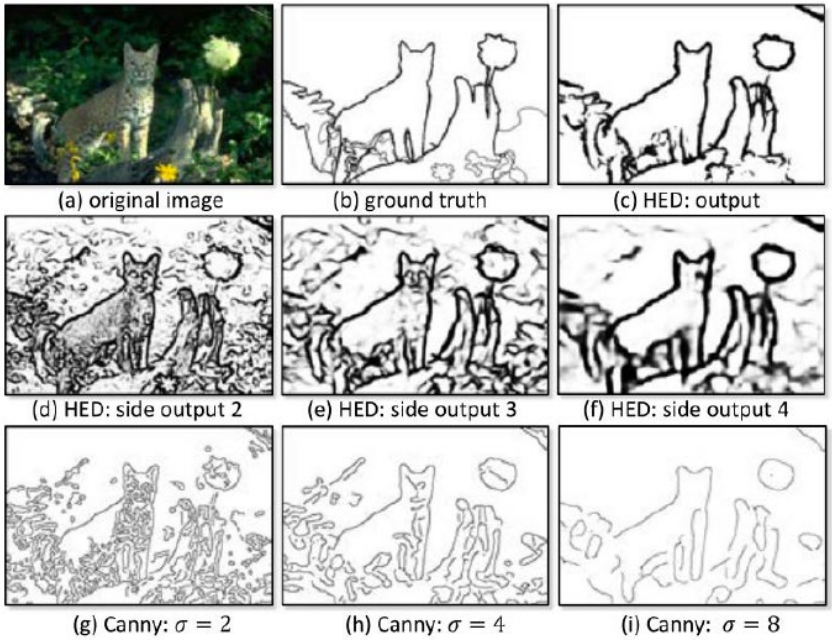
## Comparativas sobre *benchmarks*




Martin et al., *Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues*. PAMI'04

# Aprendizaje automático



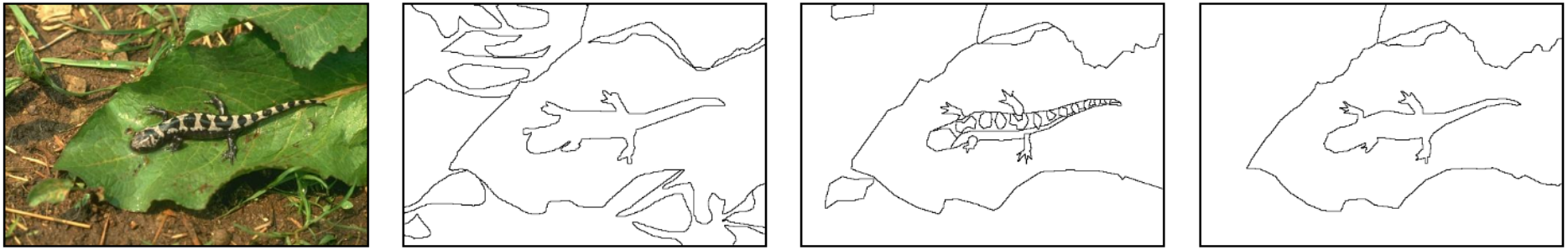




S. Xie, Z. Tu, *Holistically-Nested Edge Detection*. ICCV 2015

# Evolución

Comparativa con BSDS500

- 500 imágenes
- Anotadas por un promedio de 5 sujetos



Original Image | Subject 1 | Subject 2 | Subject 3

P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. Contour Detection and Hierarchical Image Segmentation. PAMIU 2011

# Evolución

## Comparativa con BSDS500

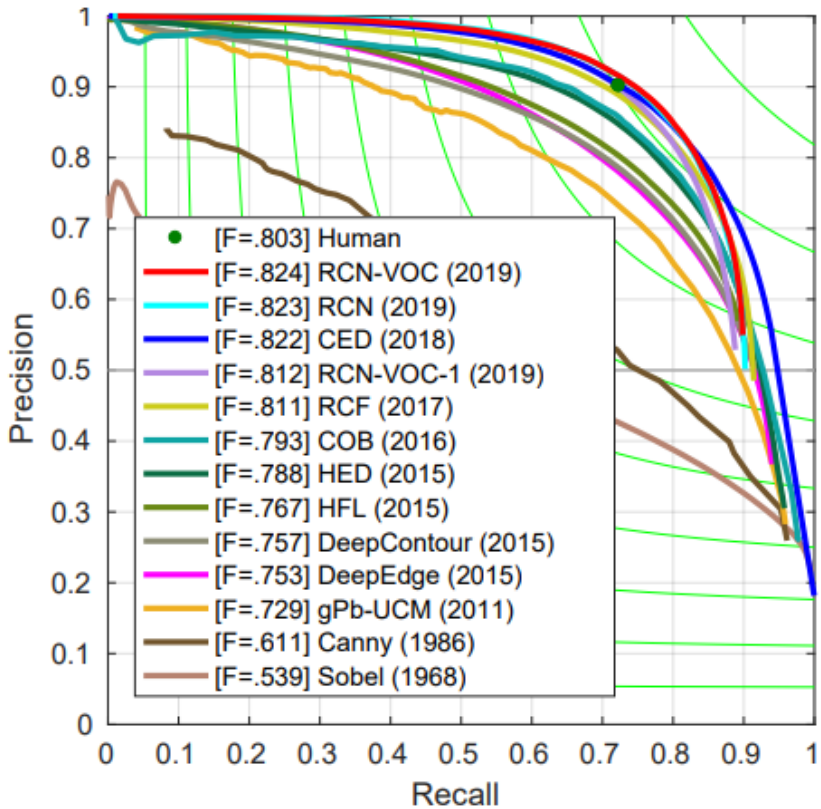


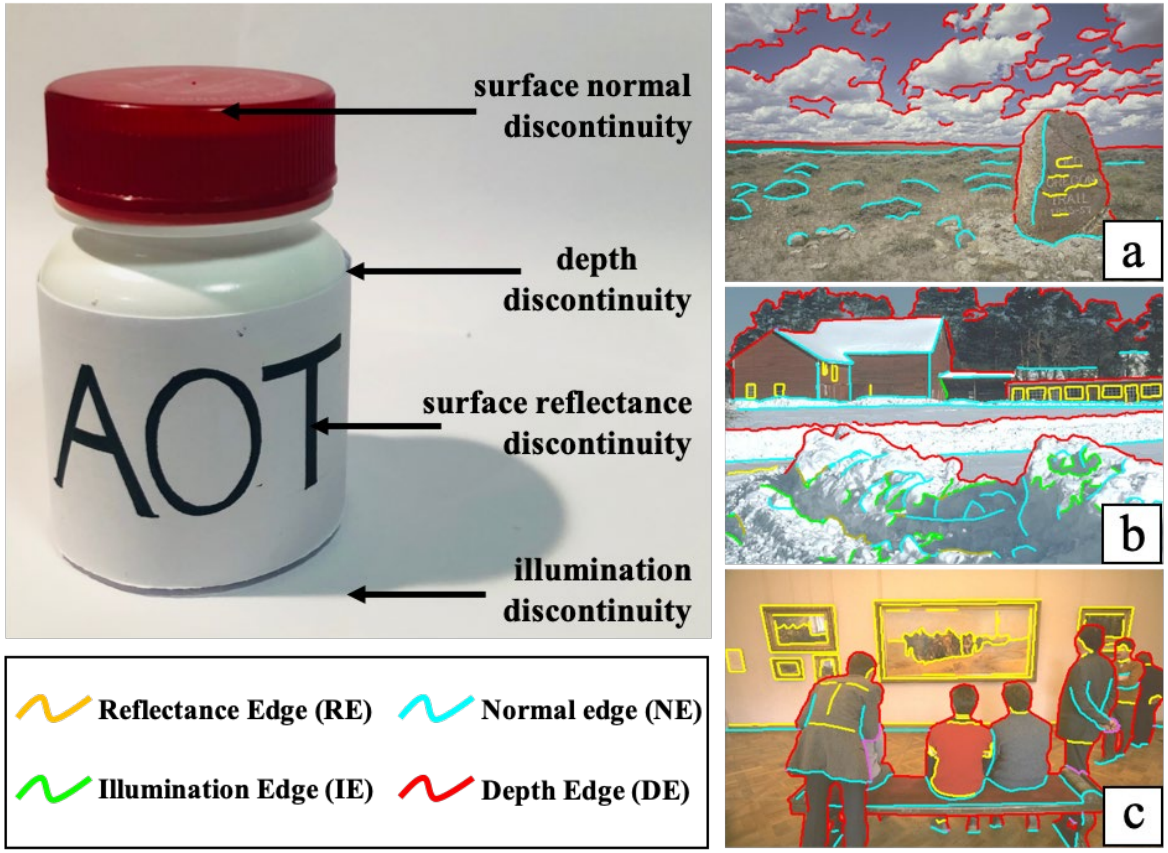

**Table 2.** Comparison on BSDS500

| Net | ODS | OIS | AP |
|---|---|---|---|
| **RCN-VOC** | **.824** | .839 | .837 |
| RCN | .823 | .838 | .853 |
| CED | .822 | **.840** | **.895** |
| RCN-VOC-1 | .812 | .827 | .822 |
| RCF | .811 | .830 | .846 |
| Human | .803 | .803 | - |
| COB | .793 | .819 | .849 |
| HED | .788 | .808 | .840 |

A. P- Kelm, V. S. Rao & U. Zölzer. Object Contour and Edge Detection with RefineContourNet; CAIP 2018

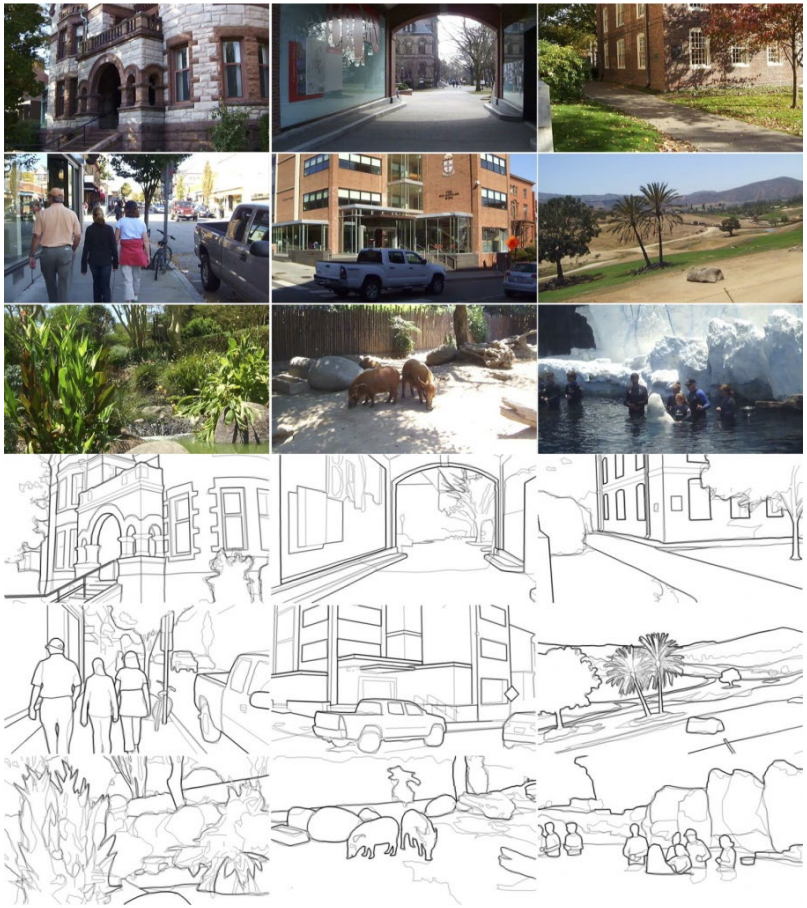
# Evolución

BSDS-RIND, a partir de BSDS500 analiza tipologías de bordes



# Evolución

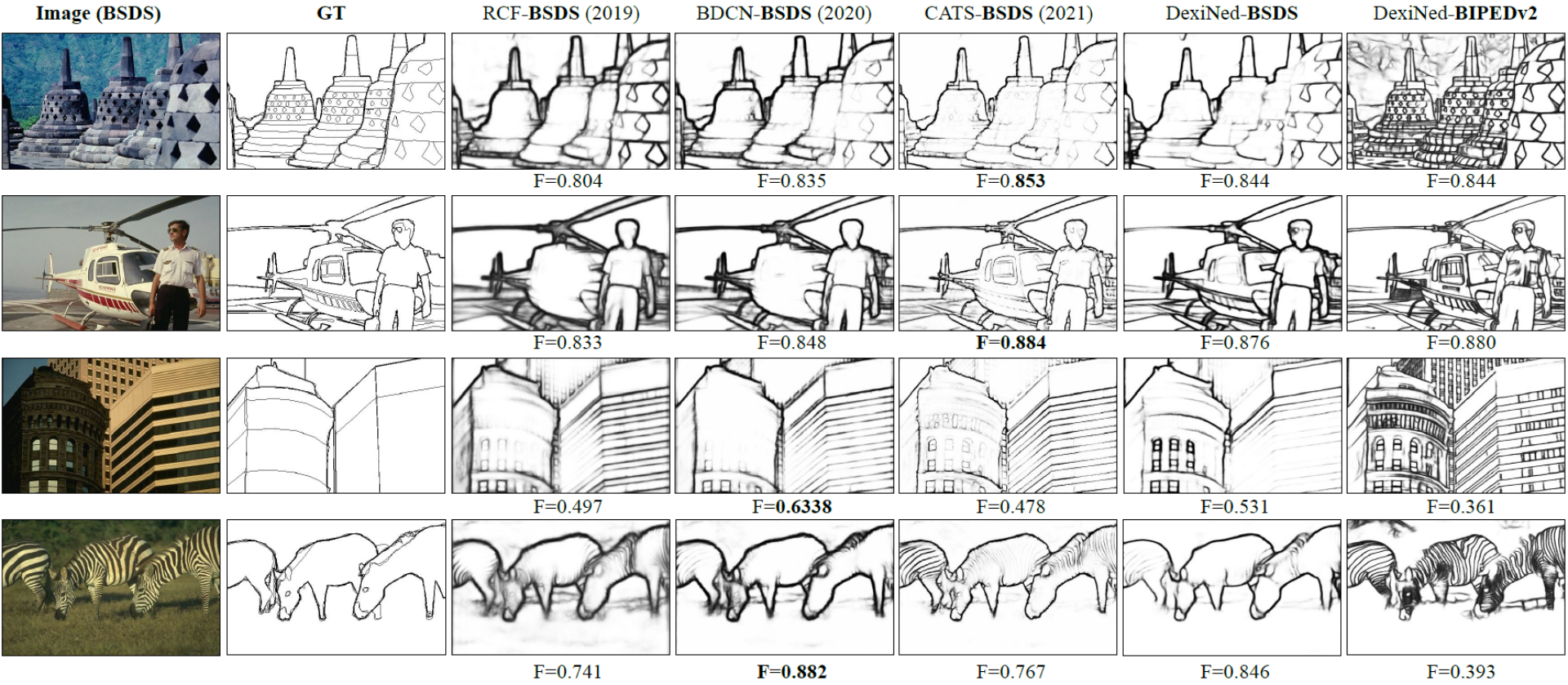
## MDBD (Multicue Dataset for Edge Detection)



# Evolución

MDBD (Multicue Dataset for Edge Detection)

[DexiNed]



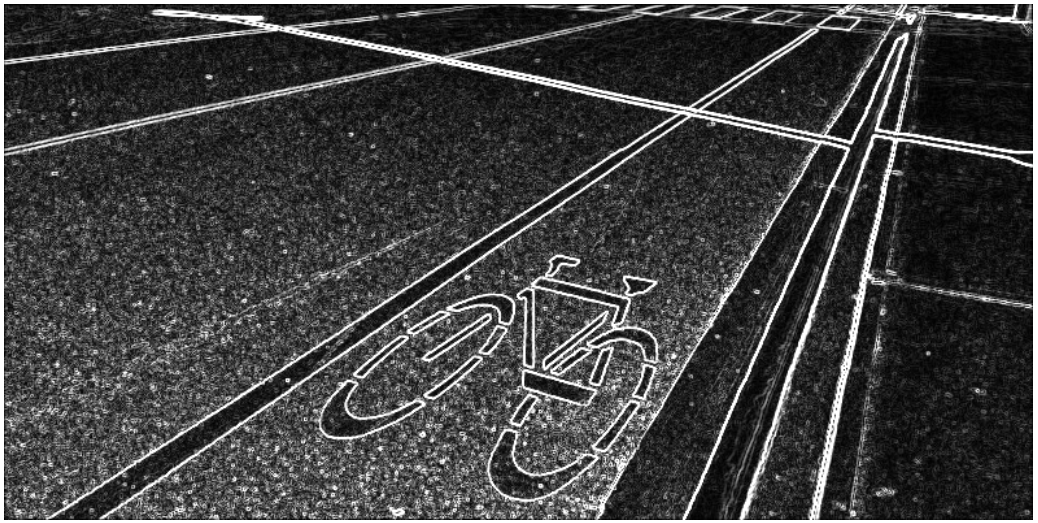
# Líneas

Bordes y curvas describen contornos de objetos

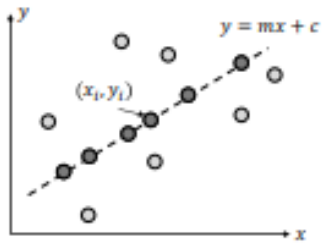
Muchos objetos artificiales están delimitados por líneas

Determinar bordes útiles, completarlos, y tolerar ruido

**Transformada de Hough** (1962)



# Líneas

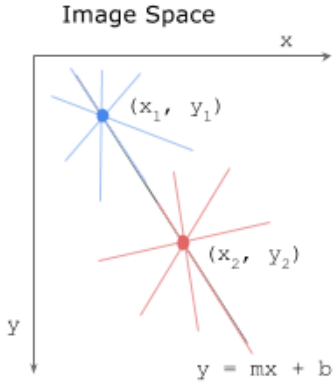



S. Nayar

Línea recta

$$y = mx + b$$

Determinar líneas presentes a partir de conjunto de puntos

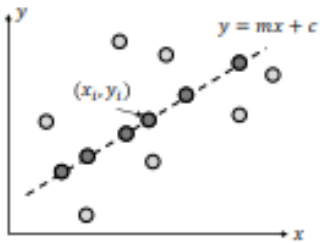



Para un punto en la imagen $(x_i,y_i)$

$$b = y_i - mx_i$$

Múltiples líneas pasan por $(x_i,y_i)$ variando $m$ y $b$

# Líneas

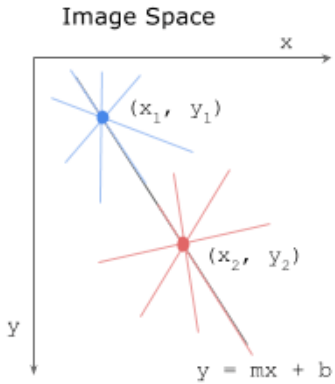



S. Nayar

Línea recta

$$y = mx + b$$

Determinar líneas presentes a partir de conjunto de puntos

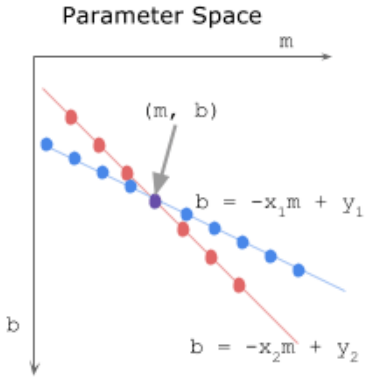



Para un punto en la imagen $(x_i,y_i)$

$$b = y_i - mx_i$$

Múltiples líneas pasan por $(x_i,y_i)$ variando $m$ y $b$

Espacio 2D $(m,b)$ líneas de cada $(x_i,y_i)$




Cuantizar espacio $(m,b)$

Esquema de votación, más votos ganan

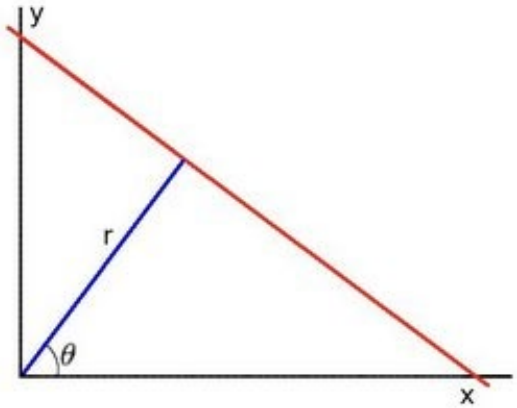
# Líneas

Inconveniente del espacio *(m,b)*, rango $0 \leq m \leq \infty$

Más práctica representación polar *(r,θ)*:

- $0 \leq \theta \leq 2\pi$
- $r = (n_x, n_y)$

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

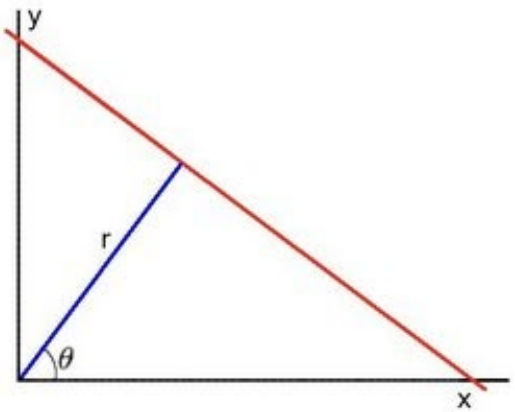


# Líneas

Inconveniente del espacio *(m,b)*, rango $0 \leq m \leq \infty$

Más práctica representación polar *(r,θ)*:

- $0 \leq \theta \leq 2\pi$
- $r = (n_x, n_y)$




Transformada de Hough orientada:

- Inicializar contadores

- Para cada segmento en *(x,y)* y $\theta = \tan^{-1} {}^{n_y}/_{n_x}$
  - calcular $r = xn_x + yn_y$
  - incrementar contador *(r,θ)*

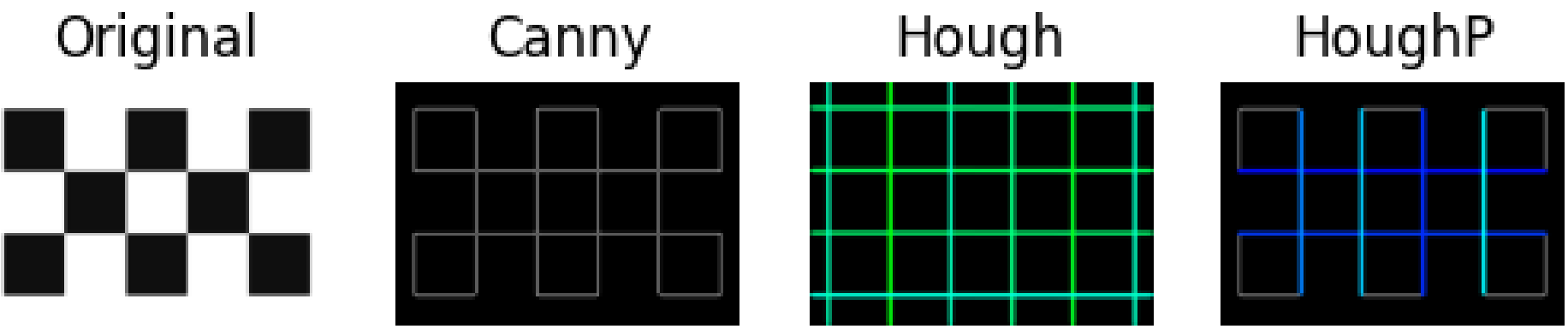
- Picos del acumulador se corresponden a líneas

# Líneas

OpenCV ofrece:

- Estándar *HoughLines* devuelve un vector de $(r,\theta)$
- Probabilística *HoughLinesP* (J.Matas et al. 2000) devuelve extremos de las rectas $(x_0,y_0,x_1,y_1)$

Canny T1 = 50 , T2 = 200

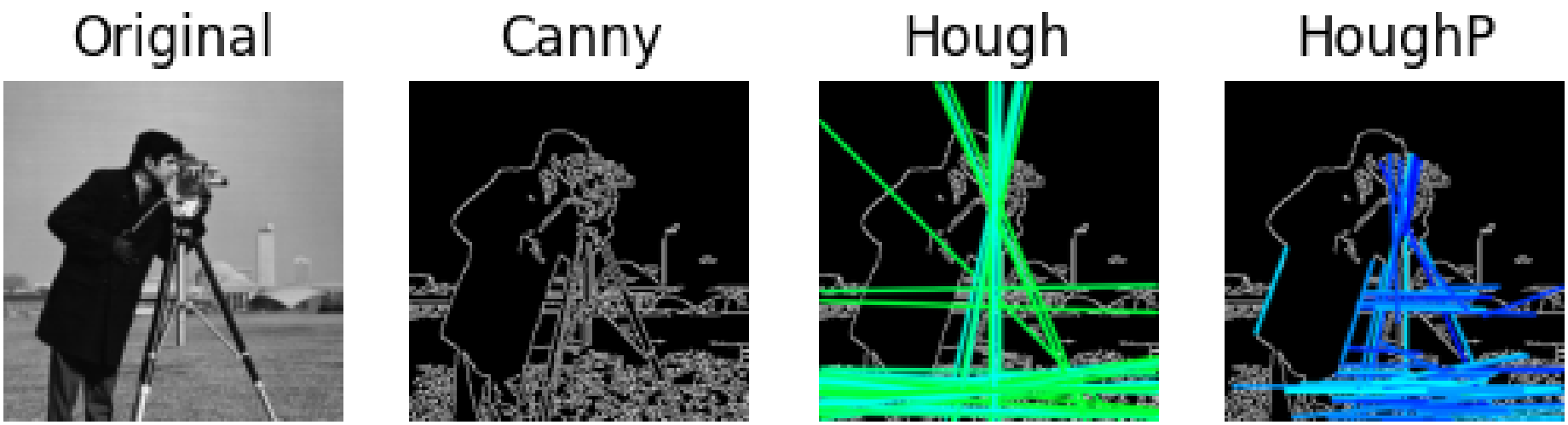


# Líneas

OpenCV ofrece:

- Estándar *HoughLines* devuelve un vector de $(r,\theta)$
- Probabilística *HoughLinesP* (J.Matas et al. 2000) devuelve extremos de las rectas $(x_0,y_0,x_1,y_1)$

Canny T1 = 50 , T2 = 200

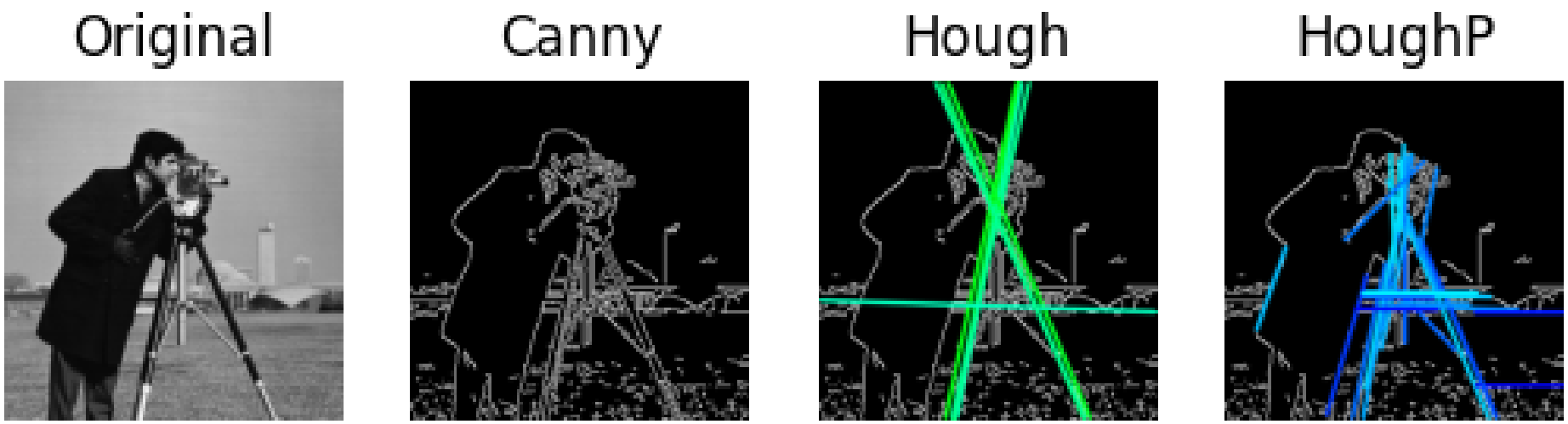


# Líneas

OpenCV ofrece:

- Estándar *HoughLines* devuelve un vector de $(r,\theta)$
- Probabilística *HoughLinesP* (J.Matas et al. 2000) devuelve extremos de las rectas $(x_0,y_0,x_1,y_1)$

Canny T1 = 150 , T2 = 200



# Líneas

OpenCV ofrece:

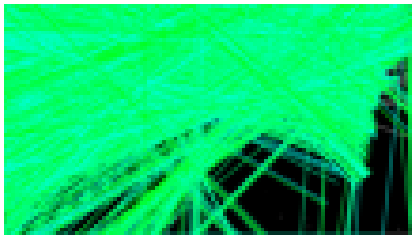
- Estándar *HoughLines* devuelve un vector de $(r,\theta)$
- Probabilística *HoughLinesP* (J.Matas et al. 2000) devuelve extremos de las rectas $(x_0,y_0,x_1,y_1)$

Canny T1 = 150 , T2 = 200

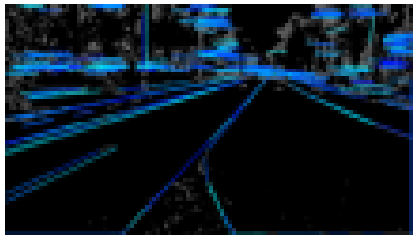
Original

Canny

Hough



HoughP

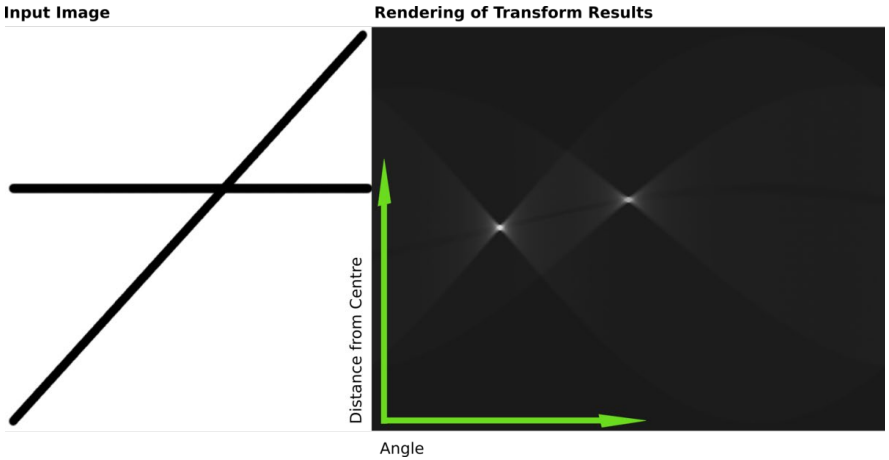


# Líneas

Ojo con discretización del espacio
Parámetros de *HoughLines*

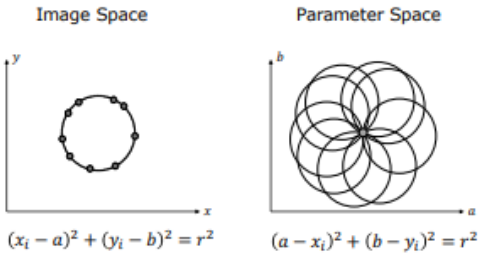
Poca resolución, se pierden líneas
Mucha resolución (celdas pequeñas) el ruido afecta

No buscar picos individuales, sino por zonas



# Líneas

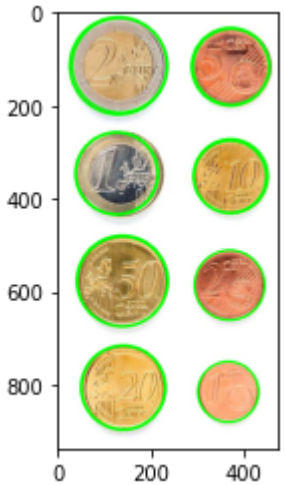
Buen funcionamiento con formas simples (pocos parámetros)
Transformada de Hough para contornos circulares
Para un *r* conocido un punto $(x_i,y_i)$ define ciertos círculos




S. Nayar

Definir rango de *r*

Existe versión generalizada de Hough



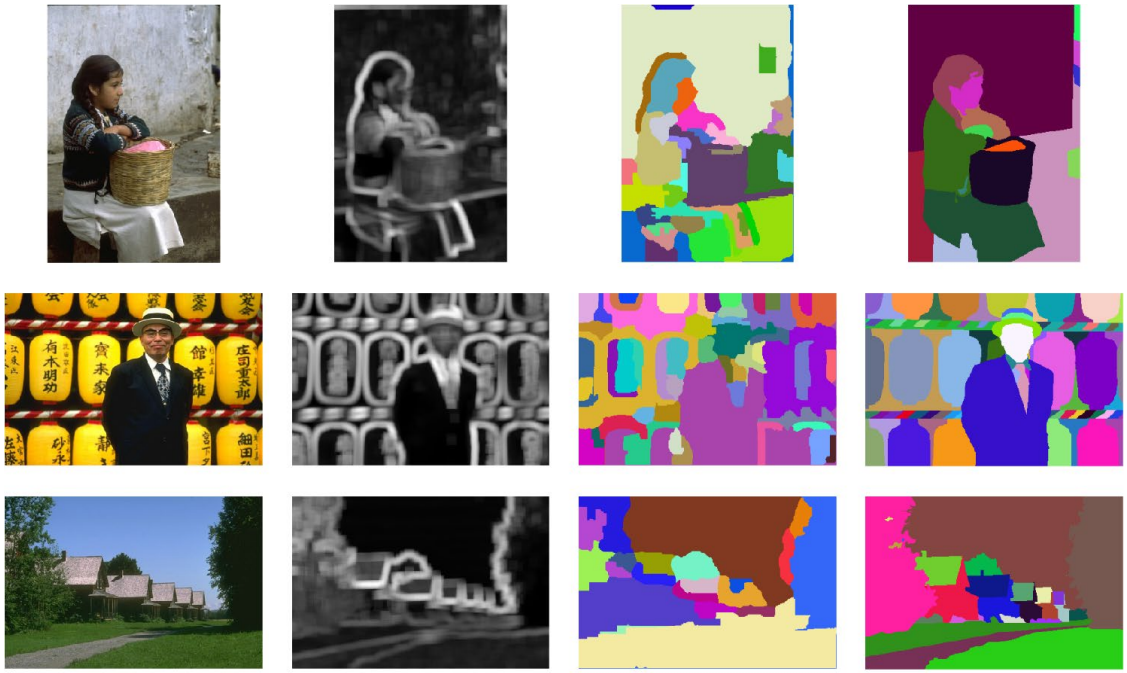
# Segmentación

Grupos de píxeles (*patches*) que comparten características
Agrupamiento o *clustering*
Información espacial, vecindad
Problema mal definido

# Segmentación

División y unión de regiones

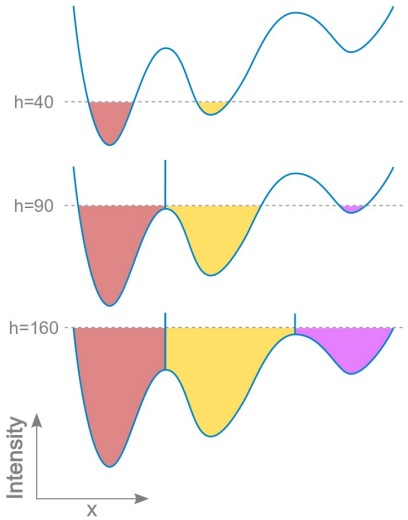
Basado en histogramas, color, textura, movimiento

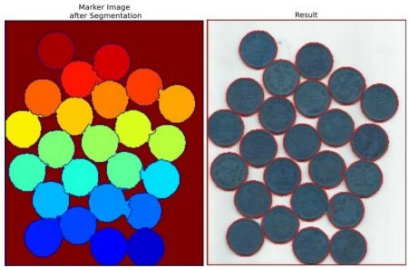


(a) (b) (c) (d)

# Segmentación

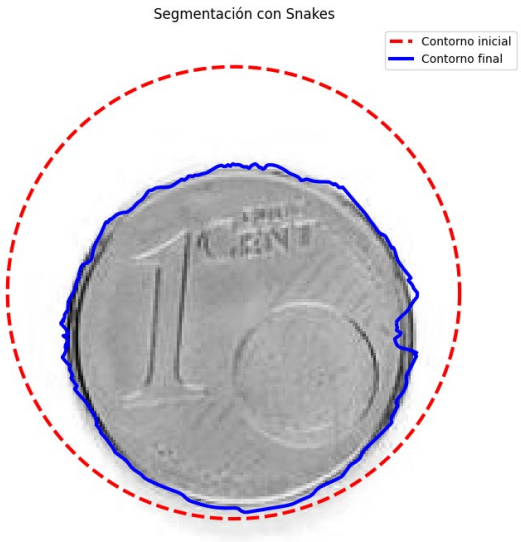
Inundación o *watershed*, imagen como perfil topográfico
Transformada distancia
Semillas en mínimos locales, se *inunda*
Fronteras en puntos de encuentro entre cuencas



# Segmentación

*Snakes*, contornos activos y funciones de energía
Disponible en [*skimage.segmentation*]



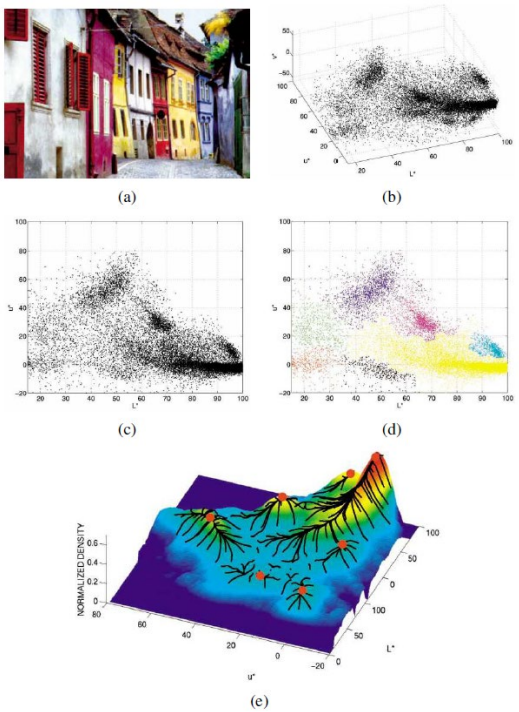
# Segmentación

*Meanshift*

Modela con *k-means* y mezcla de gausssianas

Localiza picos con centros y covarianza

Aplicado en seguimiento




Szeliski

# Segmentación

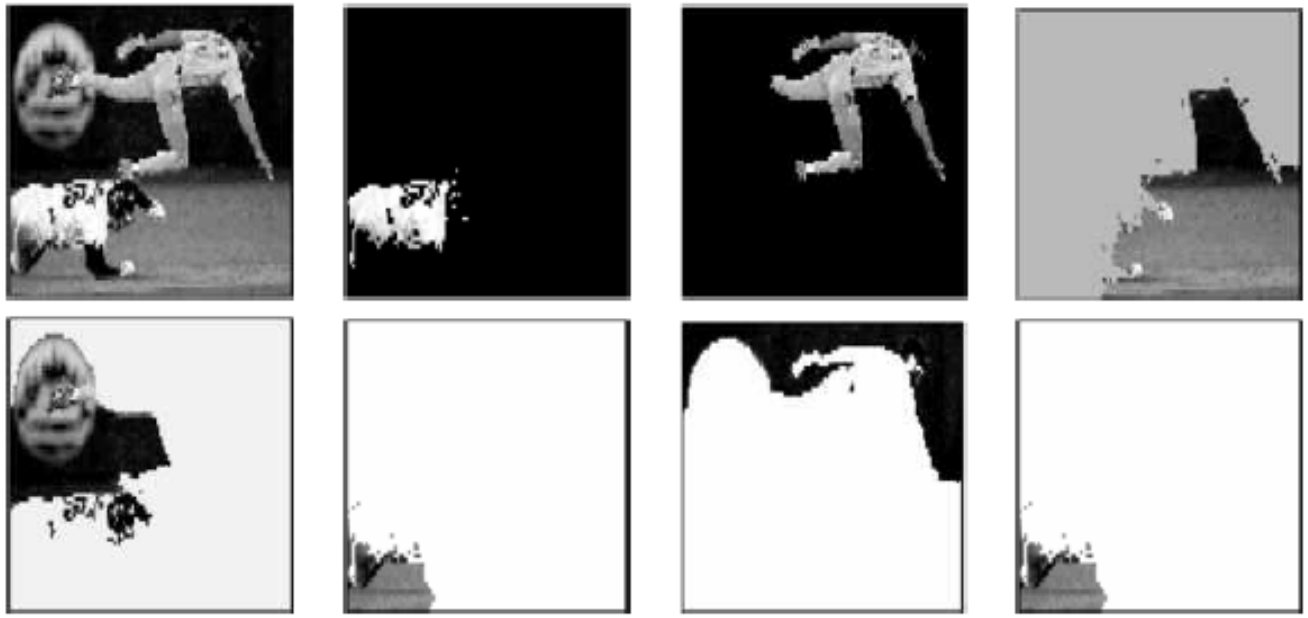
## *Meanshift*

# Segmentación

*Normalized cuts*

Afinidades entre píxeles vecinos

Separa aquellos con conexiones débiles



Szeliski

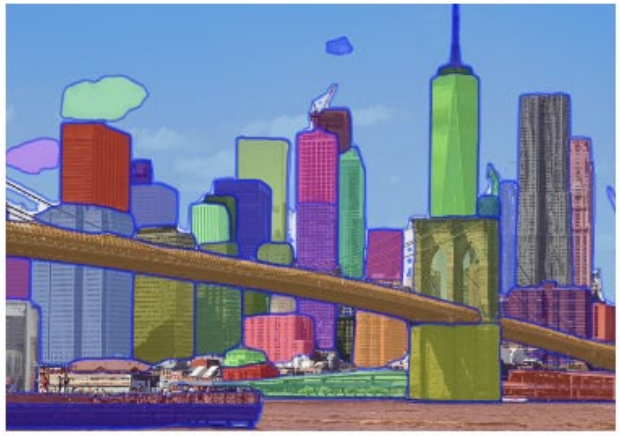
# Segmentación

SAM (*Segment Anything*) (2023)
Modelo fundamental (*Foundation Model*)
Autosupervisado, ViT
11M de imágenes, mil millones de máscaras
*Zero-shot*
SAM2, incluye vídeo

# Segmentación

SAM (*Segment Anything*) (2023)
Uso flexible
Segmentación
Dados contenedores o puntos interiores
Entrenable

Prompt it with interactive points and boxes.

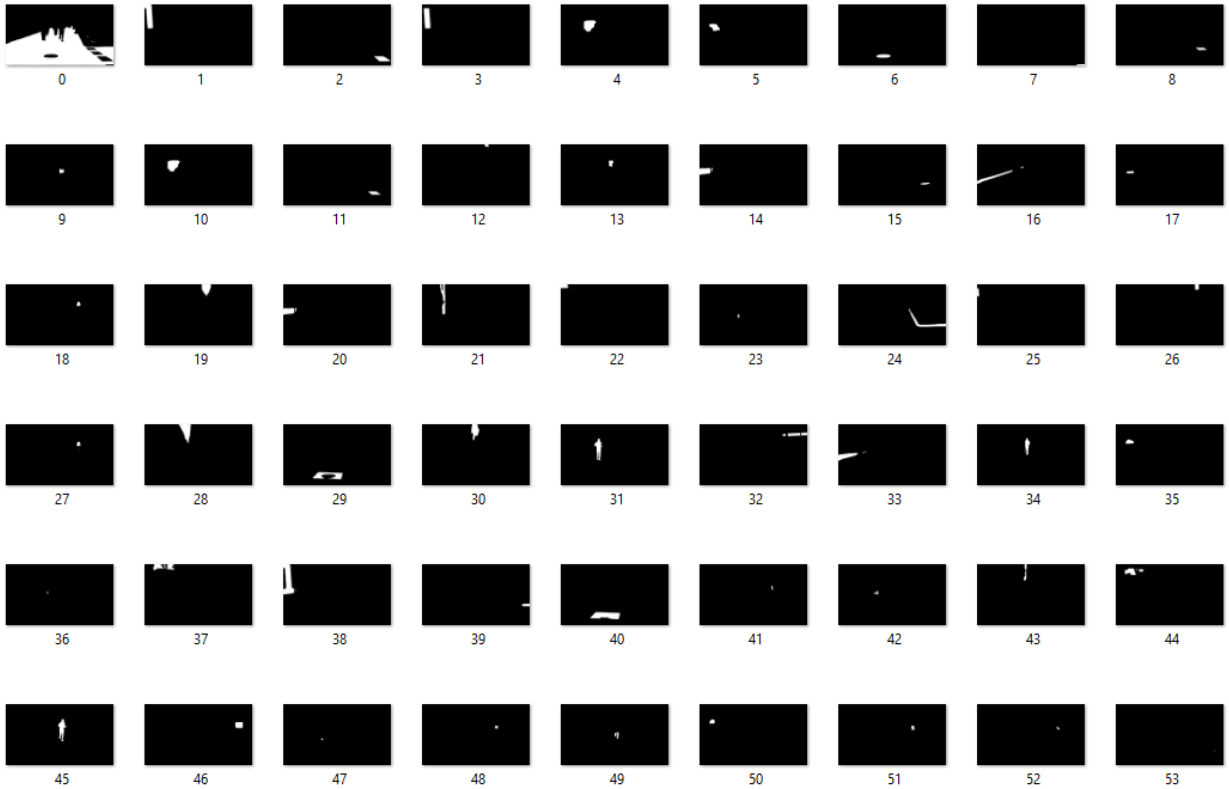


Automatically segment everything in an image.

Generate multiple valid masks for ambiguous prompts.

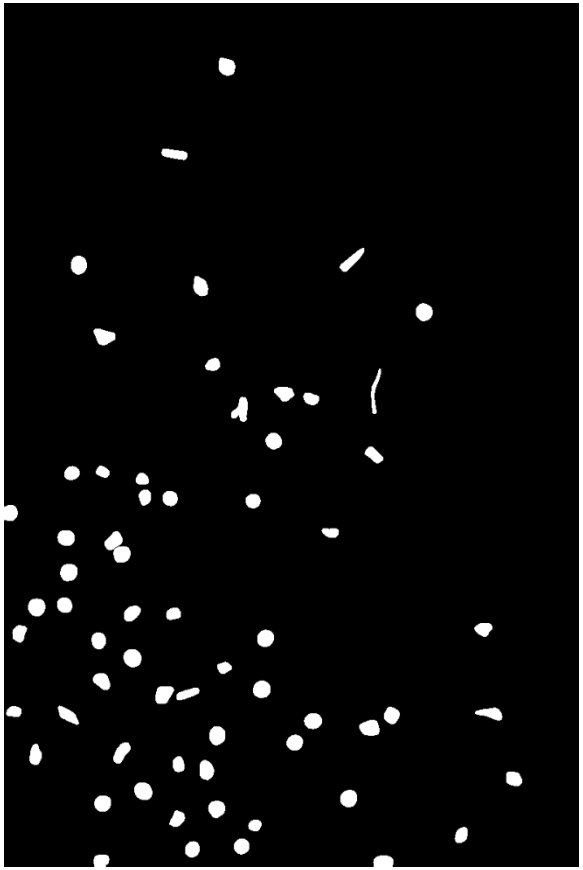
# Segmentación

SAM (*Segment Anything*) (2023)
Código y demo disponibles (no comercial)



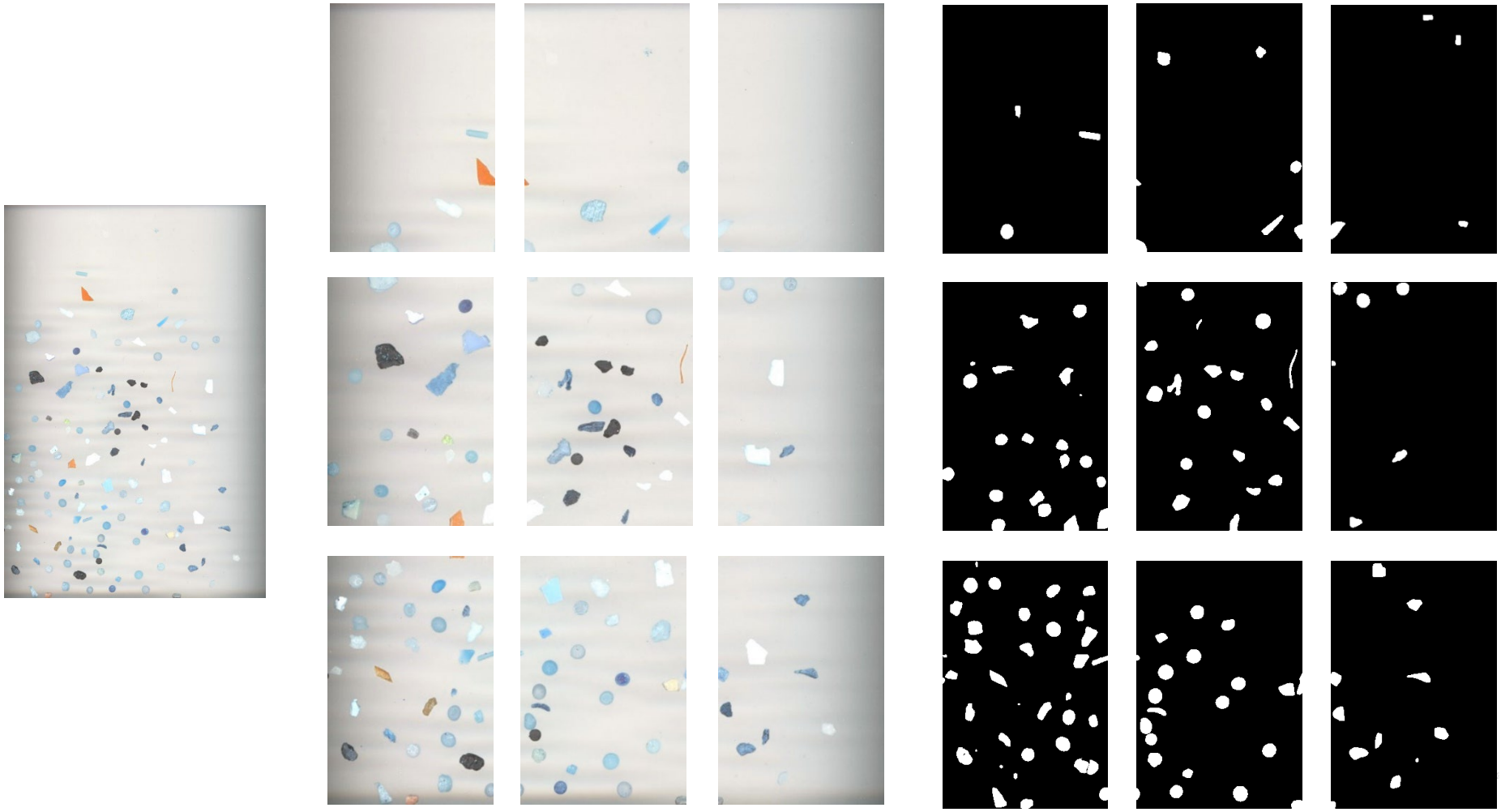
# Segmentación

SAM (*Segment Anything*) (2023)



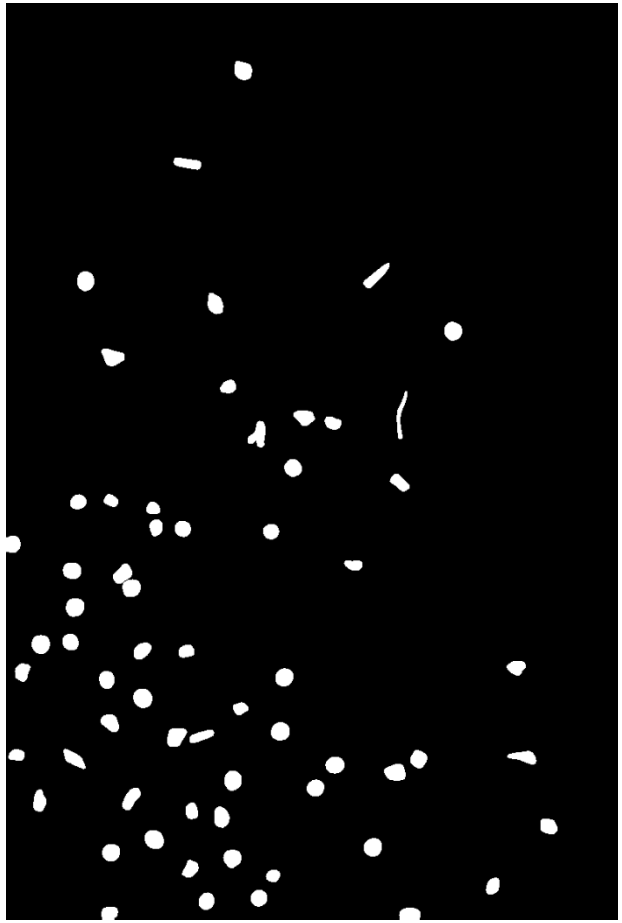
# Segmentación

SAM (*Segment Anything*) (2023)
Mosaico 3x3



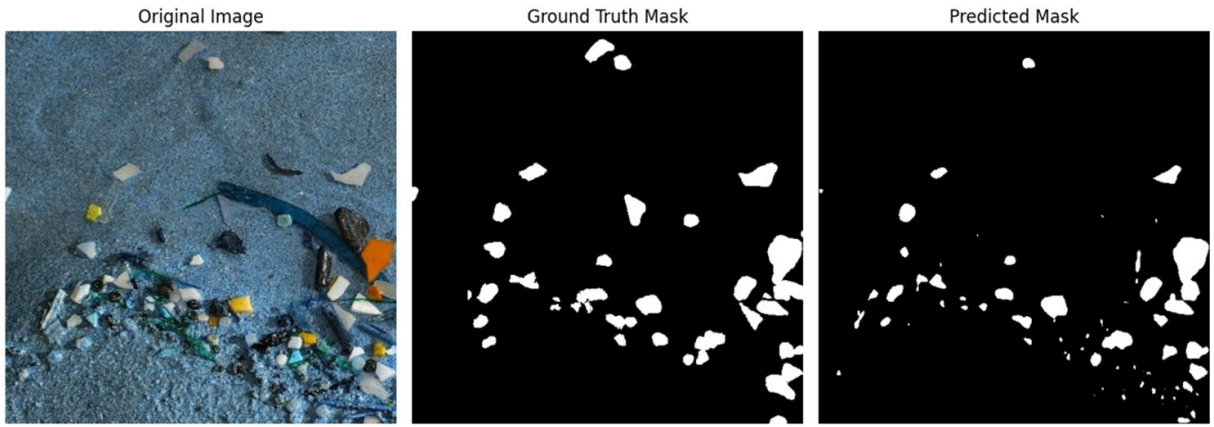
# Segmentación

SAM (*Segment Anything*) (2023)
Mosaico o no



# Segmentación

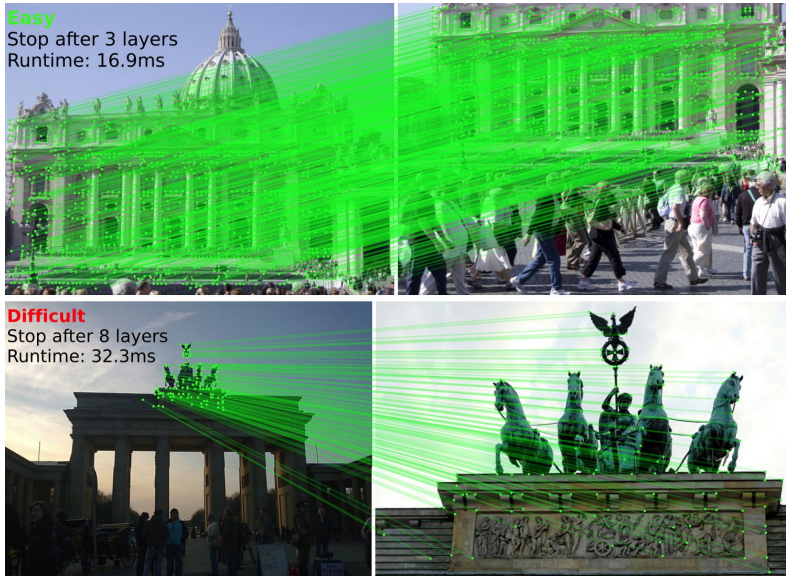
Otras técnicas integran clasificación
Aprendizaje supervisado de máscaras de determinadas clases
Unet, Oneformer, etc.

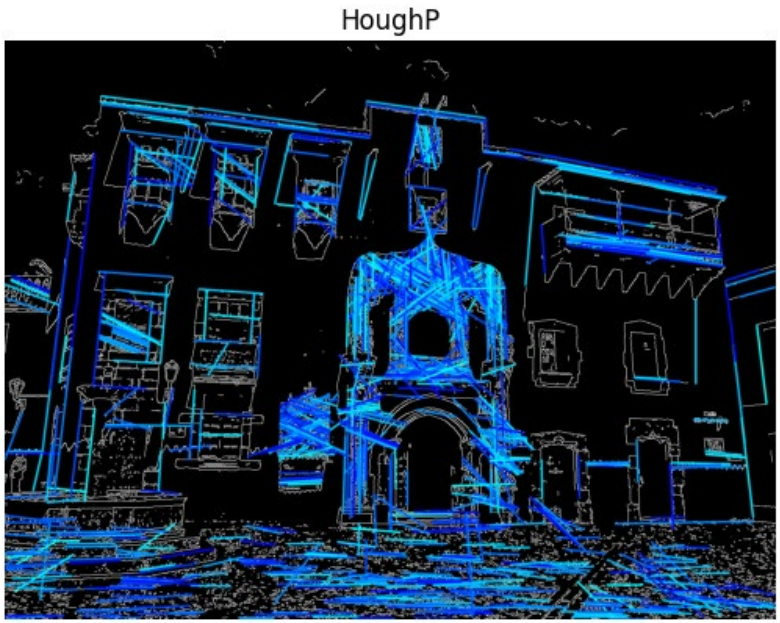


# Resumen

Puntos, líneas y *patches*

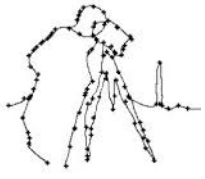
Encaje

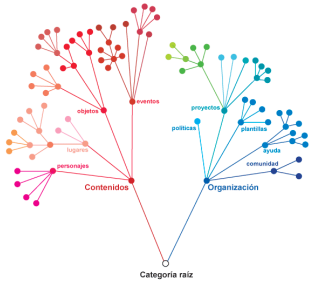


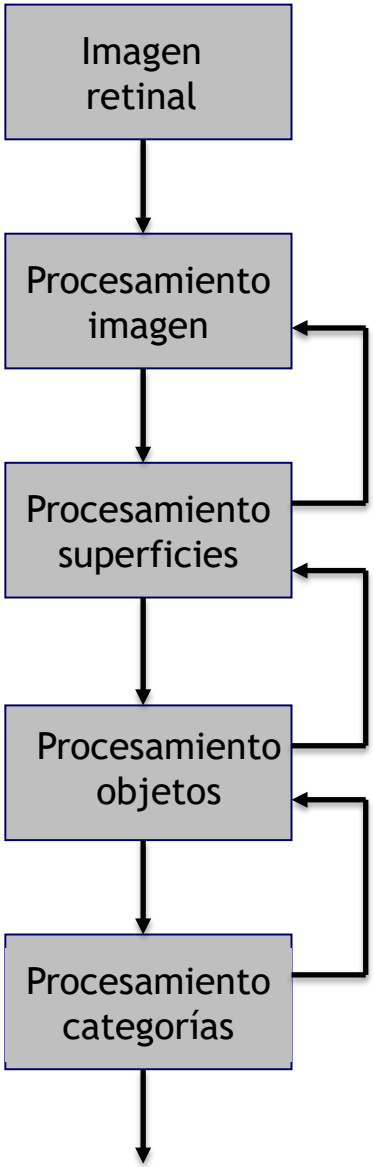



# Resumen

Niveles de D. Marr





# Bibliografía


- Shree K. Nayar, First Principles of Computer Vision
- P. Fua, Computer Vision course
- R. Klette, *Concise Computer Vision*, Springer, 2014 (online ULPGC)
- R. Szeliski. Computer Vision: Algorithms and Applications, 2nd ed. 2022
- Documentación OpenCV