Detección de puntos relevantes en imágenes

Agenda

- Motivación
- Detectores de esquinas
- Scale Invariant Feature Transform (SIFT)
- Speeded-Up Robust Features (SURF)
- Reconocimiento de objetos

Motivación

- Identificación de puntos comunes
  - Registrado (mosaicos)
  - Transformaciones
  - Estimación de movimientos
  - Cálculo de disparidad
  - Reconstrucción 3D
  - Indizado y búsqueda en bases
  - Navegación robótica
  - ...
- Buscamos *características* relevantes, *fáciles* (visibles, robustas, ...) de *perseguir*.
- Esquinas (*corners*) suelen ser candidatos.

Motivación

- Identificación de puntos comunes
  - Registrado (mosaicos)
  - Transformaciones
  - Estimación de movimientos
  - Cálculo de disparidad
  - Reconstrucción 3D
  - Indizado y búsqueda en bases
  - Navegación robótica
  - ...
- Buscamos *características* relevantes, *fáciles* (visibles, robustas, ...) de *perseguir*.
- Esquinas (*corners*) suelen ser candidatos.

Puntos relevantes obtenidos con *Harris*.

Motivación

- Detectar
  - Determinar una *propiedad* que los identifica (derivadas, curvaturas, ...).
  - Coincidan en ambas imágenes, búsqueda por separado.
  - Robusto.
- Describir
  - Extraer un conjunto de valores (descriptores) que lo representan.

$$v_1(i) = (d_1^1(i), d_2^1(i), \ldots, d_m^1(i))$$

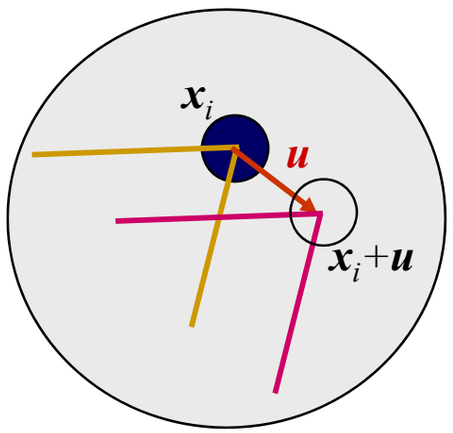
- Emparejar (*matchear*)
  - Comparar los descriptores para encontrar parejas de puntos.

$$f_{\text{dist}}(v_1, v_2)$$

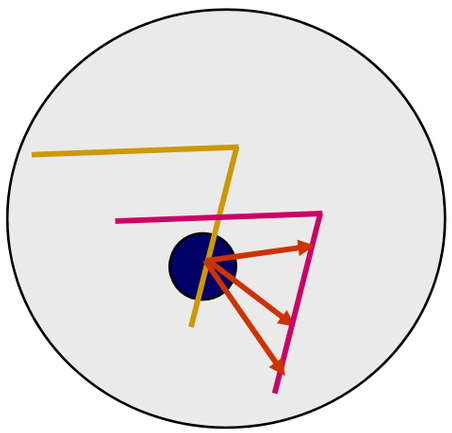
Puntos relevantes obtenidos con *Harris*.

¿Cómo buscar estos puntos relevantes?

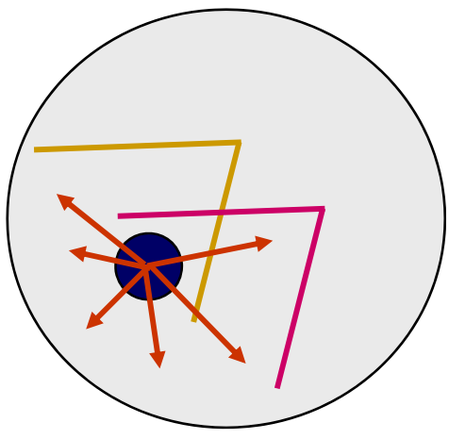
- ¿Qué tipo de *características* deberían tener estos *puntos* y sus correspondientes?
  - Dos aproximaciones:
    - Identificar *características* locales en una imagen identificables con exactitud en otra imagen: imágenes similares (p.e. video).
    - Identificar *puntos* independientemente en cada imagen y luego encontrar parejas basado en *características* locales: imágenes con cambios de apariencia o vistas.
- Deben tolerar cierta invarianza a diferencias geométricas y fotométricas entre las imágenes.



Esquinas (estable)

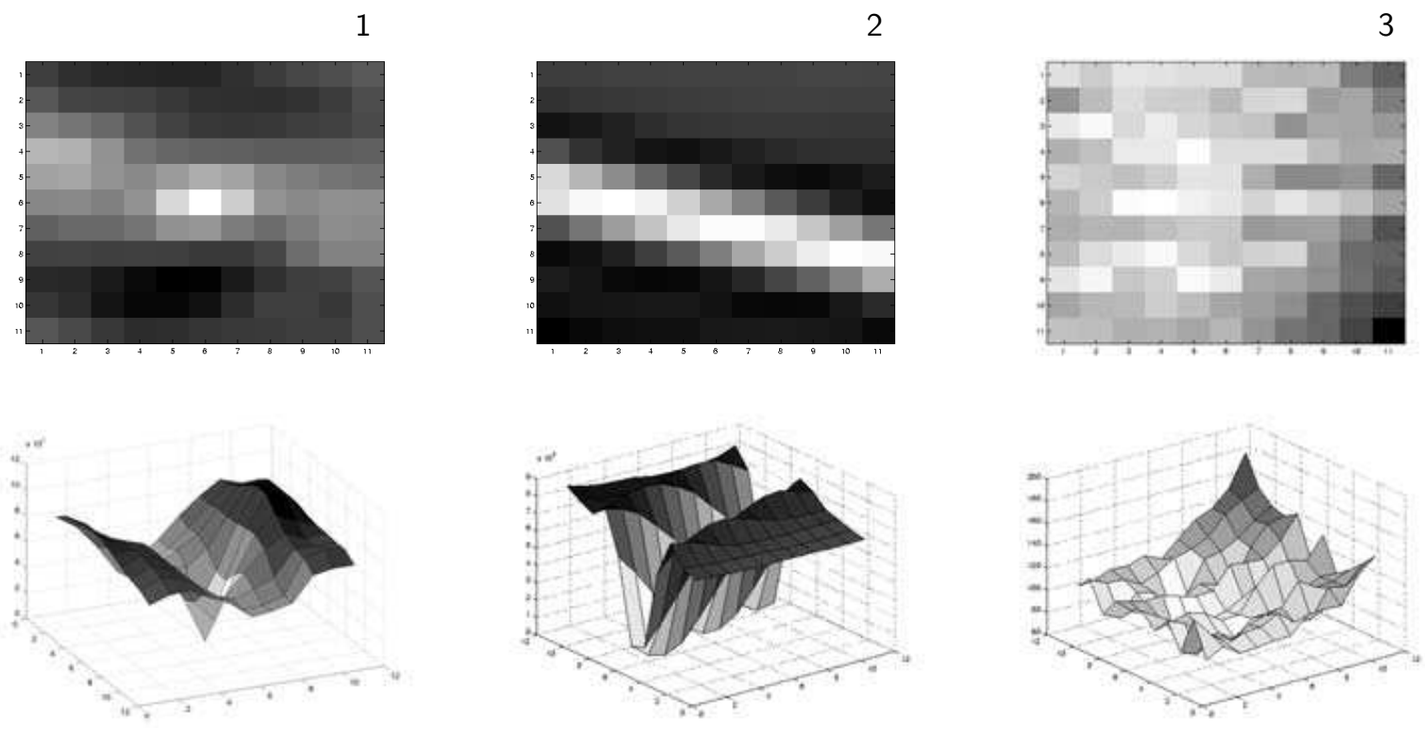


Bordes: varios candidatos

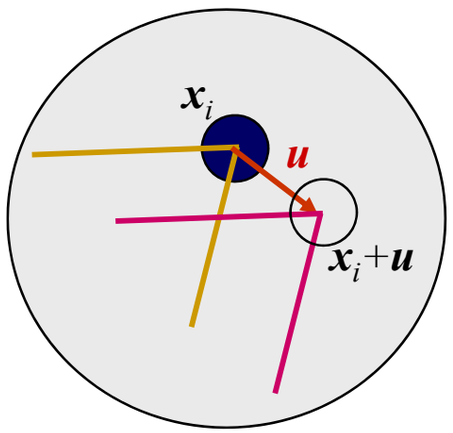


Regiones sin textura: demasiados candidatos

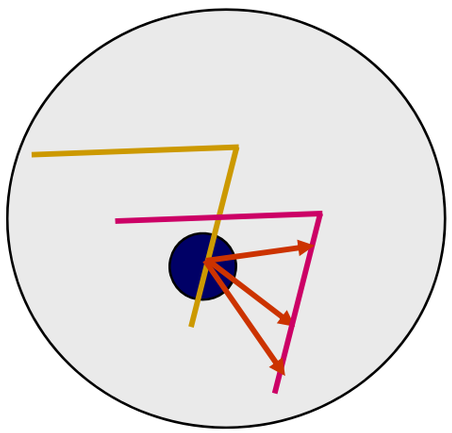
¿Cómo buscar estos puntos relevantes?



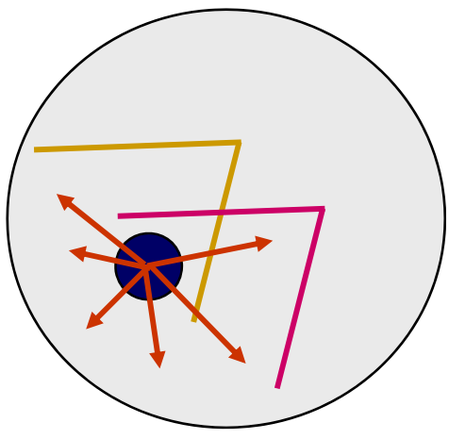
Auto-correlación de los patches marcados en la imagen a la izquierda.



Esquinas (estable)



Bordes: varios candidatos



Regiones sin textura: demasiados candidatos

¿Cómo buscar estos puntos relevantes?

$$E_{\text{WSSD}} = \sum_i w(\mathbf{x}_i) \left(I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)\right)^2$$

ventana centrada en $\mathbf{x}_i$

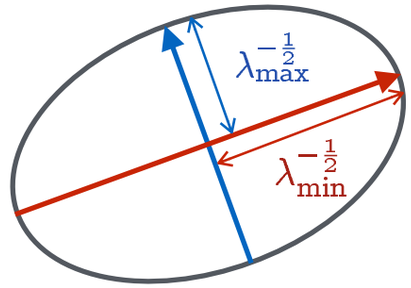
$$E_{\text{AC}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i) \left(I_0(\mathbf{x}_i + \Delta_{\mathbf{u}}) - I_0(\mathbf{x}_i)\right)^2$$

pequeñas variaciones locales

$$\begin{aligned}
E_{\text{AC}}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{x}_i) \left(I_0(\mathbf{x}_i + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)\right)^2 \\
&\approx \sum_i w(\mathbf{x}_i) \left(I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i)\Delta\mathbf{u} - I_0(\mathbf{x}_i)\right)^2 \qquad \text{Taylor} \\
&= \sum_i w(\mathbf{x}_i) \left(\nabla I_0(\mathbf{x}_i)\Delta\mathbf{u}\right)^2 \\
&= \Delta\mathbf{u}^\top \mathbf{A} \Delta\mathbf{u}
\end{aligned}$$

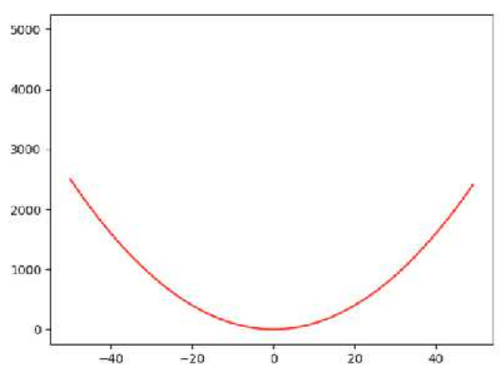
$$\nabla I_0(\mathbf{x}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{x}_i)$$

Matriz autocorrelación: $\mathbf{A} = w \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$ con valores propios $(\lambda_{\max}, \lambda_{\min})$

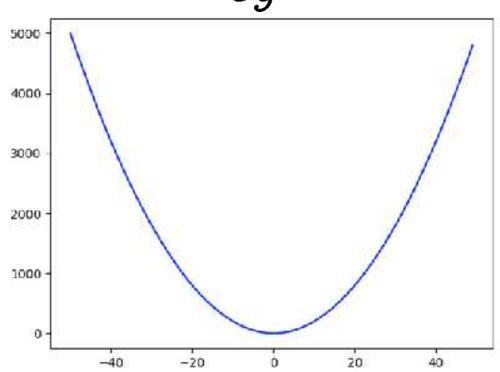


$$E_{AC} = ax^2 + bxy + cy^2$$

ax^2

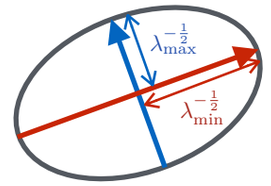


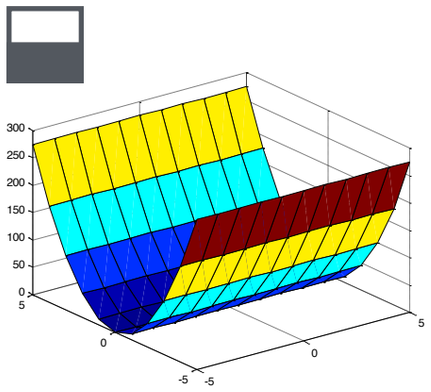
cy^2



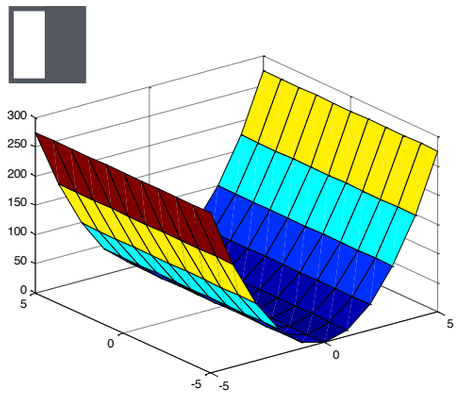
b?

¿Cómo buscar estos puntos relevantes?

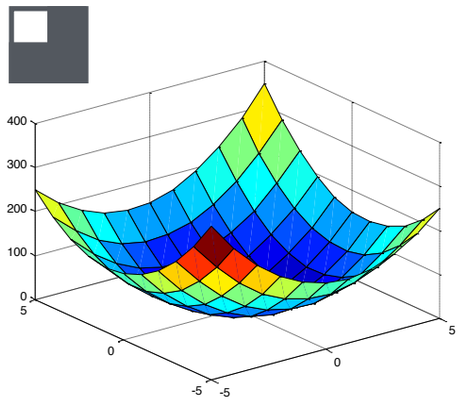




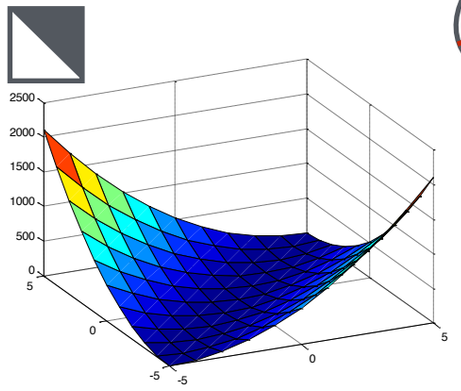
$\lambda_{\max} \gg \lambda_{\min} = 0$



$\lambda_{\max} \gg \lambda_{\min} = 0$



$\lambda_{\max} \approx \lambda_{\min}$



$\lambda_{\max} \gg \lambda_{\min} \approx 0$

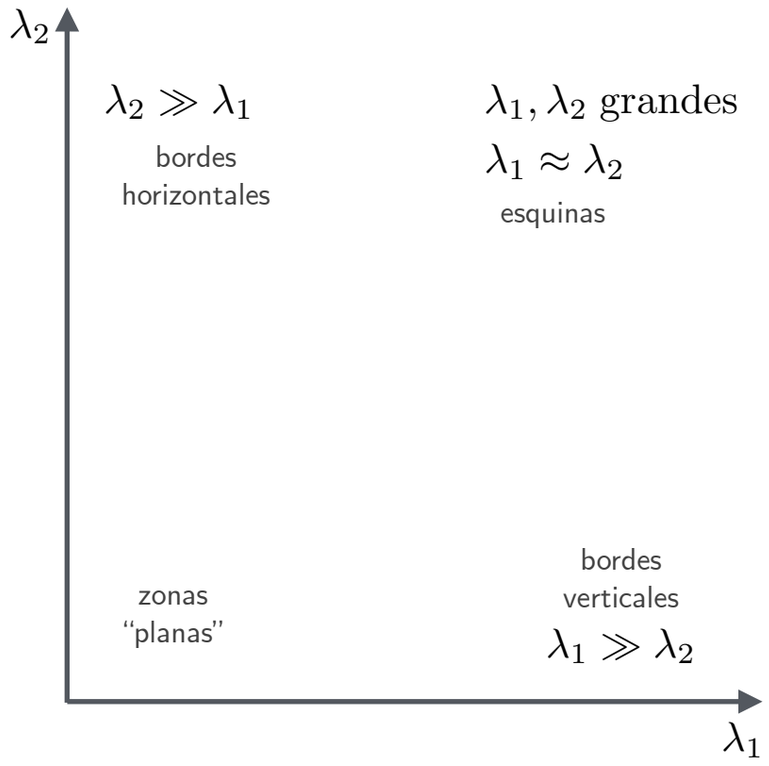
- En general si ambos valores propios son grandes tenemos un *buen* candidato. Si uno (el menor) es muy bajo es similar a un borde.
- Conviene analizar los valores propios de $\mathbf{A}$, pero puede ser costoso.
- Shi y Tomasi (1994), se basan en el valor del mínimo para encontrar los *good features to track*.

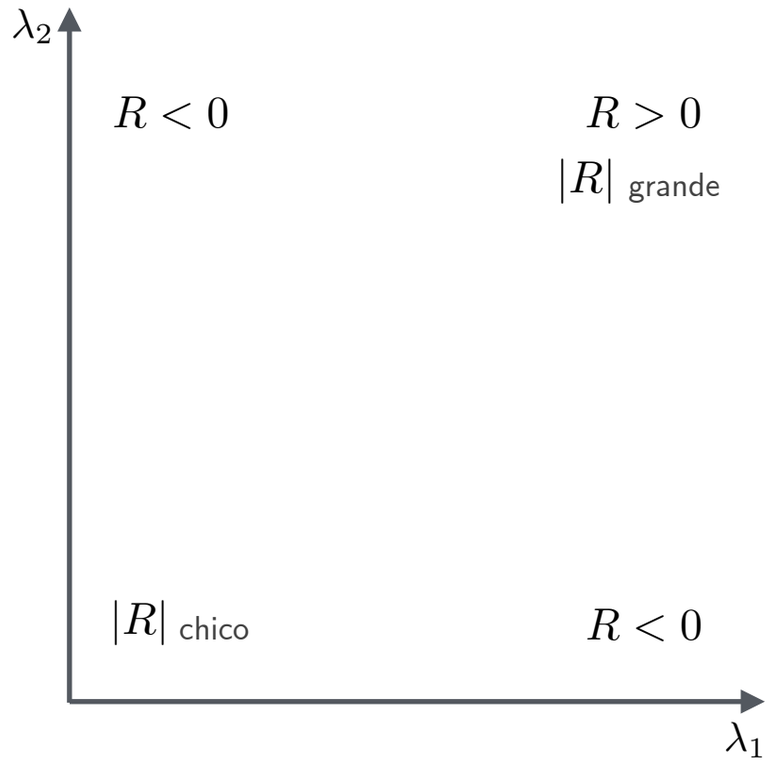
Detector de esquinas de Harris

- Harris y Stephens (1988) proponen evaluar la relación

$$\begin{aligned} R &= \det(\mathbf{A}) - k\,\text{traza}(\mathbf{A})^2 = I_x^2 I_y^2 - I_{xy}^2 - k(I_x + I_y)^2 \\ &= \lambda_{\max}\lambda_{\min} - k(\lambda_{\max} + \lambda_{\min})^2 \end{aligned}$$

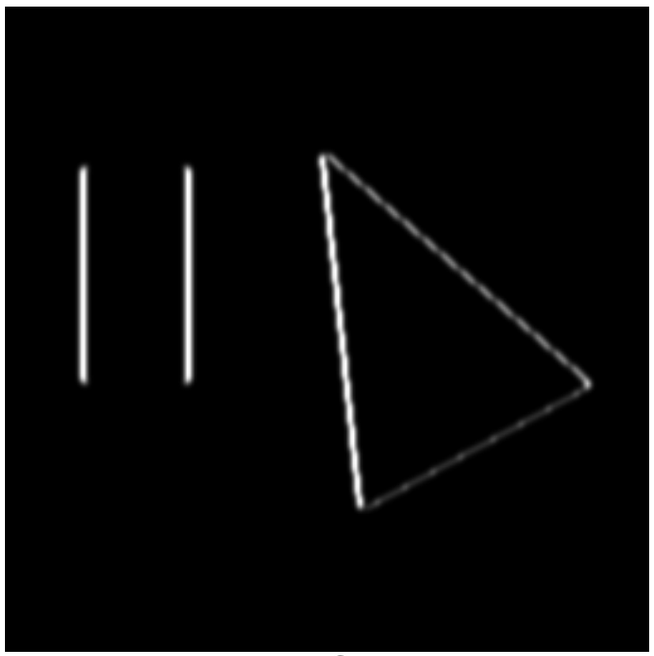
  - No es necesario calcular los valores propios



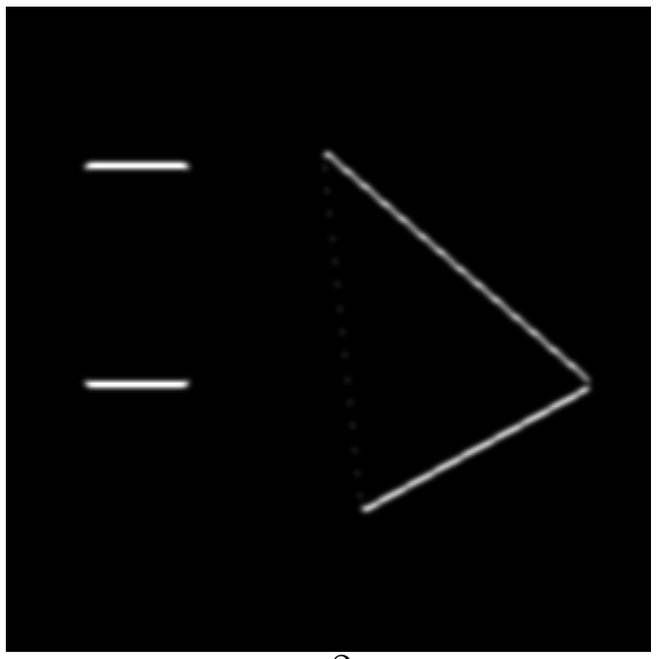


Detector de esquinas de Harris

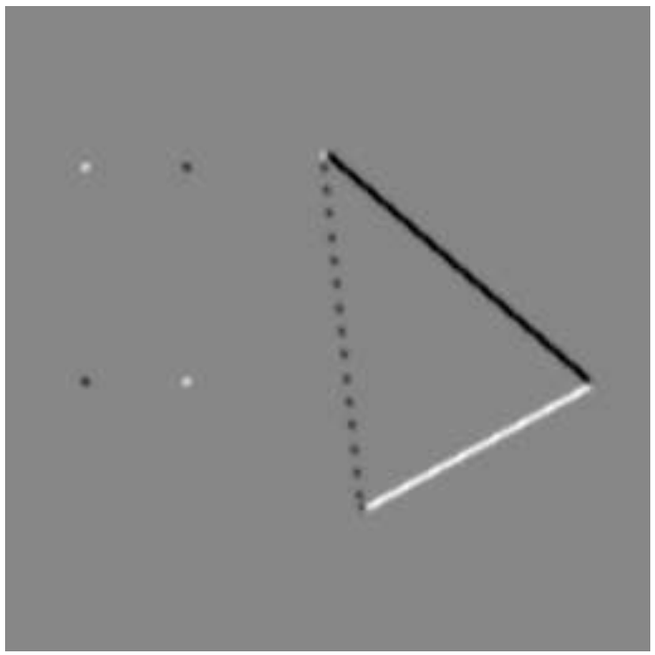
Imagen original



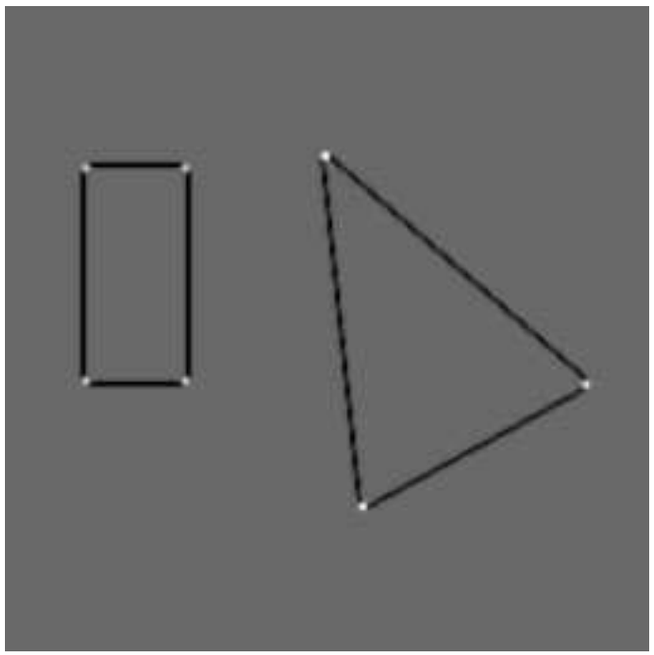
I_x^2



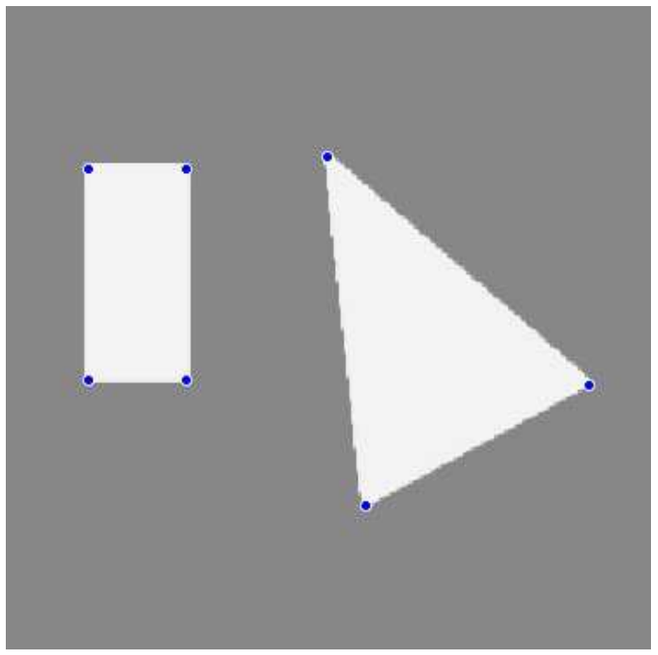
I_y^2



I_{xy}



R



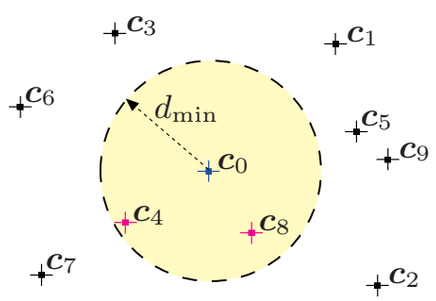
Esquinas detectadas

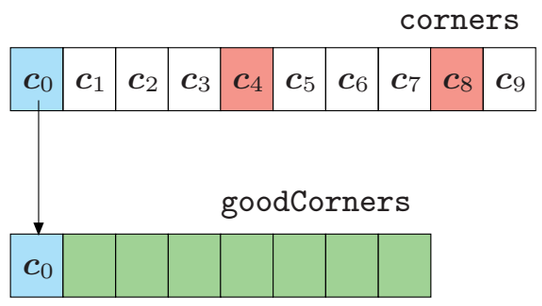
Detector de esquinas de Harris

- Algoritmo de Harris-Stephens
    1. Calcular las derivadas horizontales y verticales de la imagen.
    2. Calcular los tres términos de la matriz $\mathbf{A}$ (simétrica).
    3. Convolucionar estas imágenes con una Gaussiana de mayor varianza.
    4. Calcular la respuesta escalar R en cada pixel.
    5. Buscar máximos locales superiores a un umbral y reportarlos.

Detector de esquinas de Harris

- Detalles de implementación a tener en cuenta:
  - Filtrado de las imágenes para tomar las derivadas.
  - Las derivadas se pueden calcular con Sobel. $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - $k = 0.04$ es sugerido en el artículo.
  - Se buscan máximos locales o se umbraliza la respuesta.
    - Se puede evaluar también el valor relativo entre puntos.
  - Se eliminan los candidatos que cercanos ($d_{\min}$) a un candidato *más fuerte.*





Detector de esquinas de Harris

Imagen original

$d_{\min} = 1$

$d_{\min} = 5$

$d_{\min} = 10$

Otras métricas

$R = \lambda_{\text{min}}$ Shi y Tomasi (1994)

$R = \lambda_{\text{min}} - \alpha\lambda_{\text{max}}$ Triggs (2004)

$$R = \frac{\det(\mathbf{A})}{\text{traza}(\mathbf{A})} = \frac{\lambda_{\text{max}}\lambda_{\text{min}}}{\lambda_{\text{max}} + \lambda_{\text{min}}}$$ (Harmonic mean) Brown, Szeliski y Winder (2005)

Otras métricas

Imagen original

Candidatos de Shi-Tomasi (xg), Harris (+r),
Harris OpenCV (.b), Brown-Szeliski-Winder (sc)

Scale Invariant Feature Transform (SIFT)

- Detector y descriptor de características relevantes.
- Ampliamente utilizado (patentado para uso comercial).
- Invariante a algunas escalas, rotaciones, cambios de iluminación, punto de vista, entre otras.
- Además de la detección asocia a cada punto una huella para identificarlo en otras imágenes o vistas.
- Basado en espacio de escalas y en conceptos del sistema visual humano.

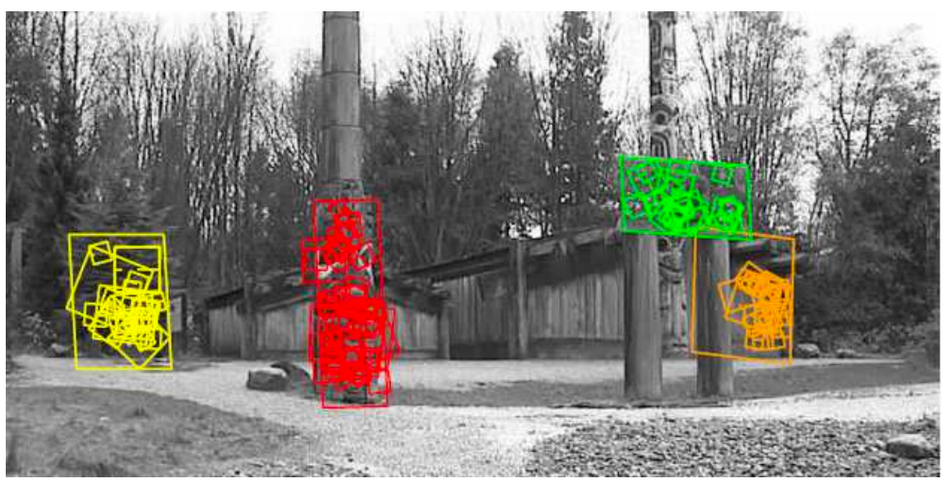
Distinctive Image Features from Scale-Invariant Keypoints

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

January 5, 2004

Abstract

This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

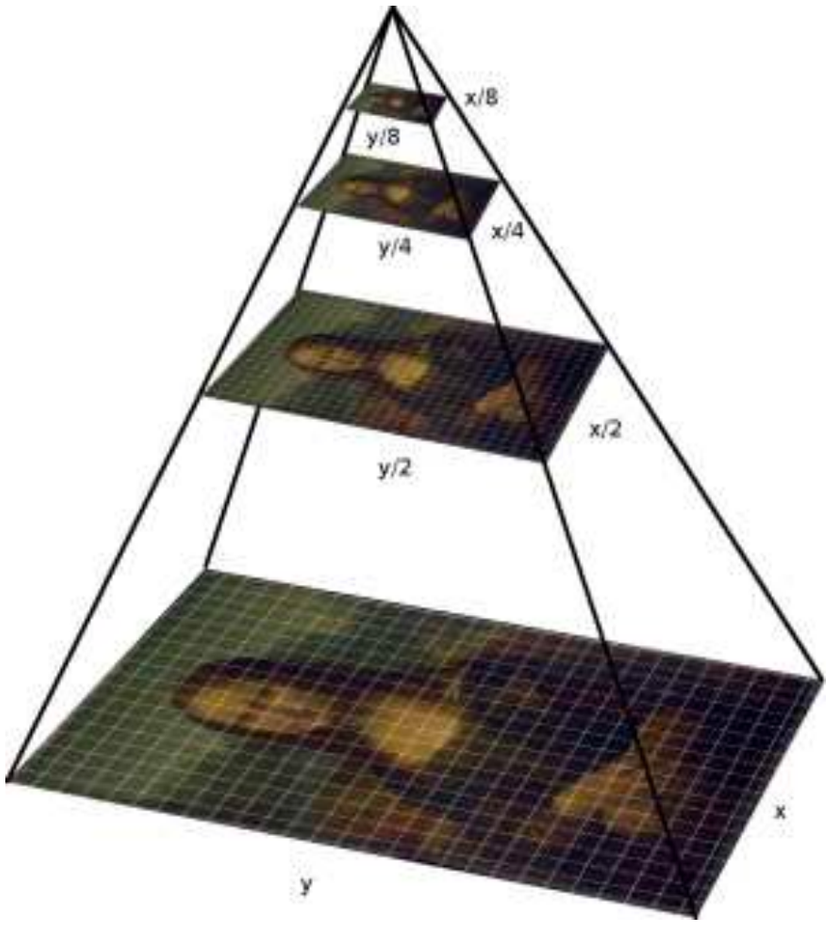


Scale Invariant Feature Transform (SIFT)

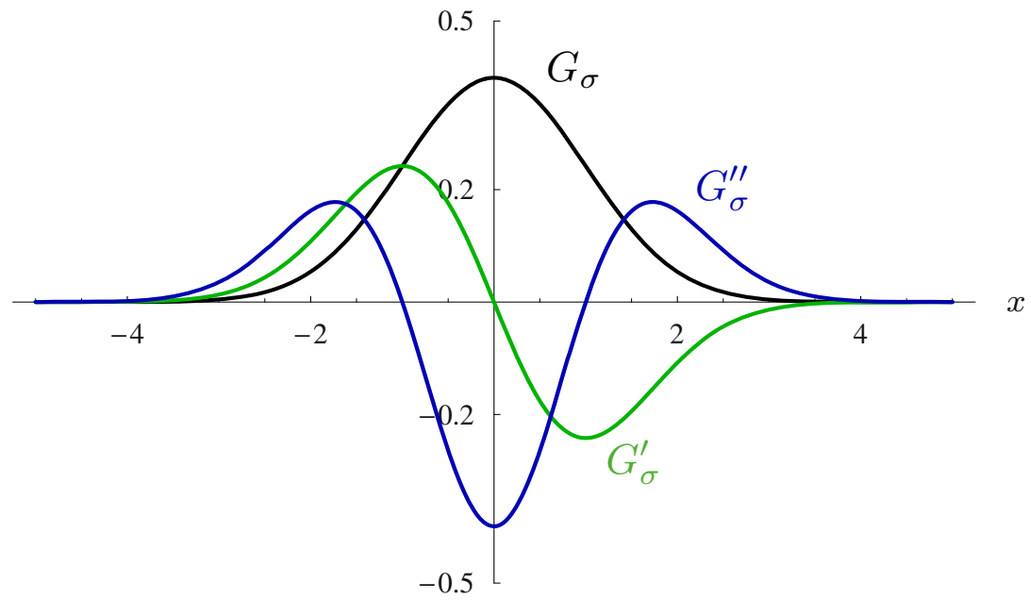
- Principales pasos en el cálculo de SIFT:
  - Detección de puntos extremos en un *espacio de escalas* obtenido con el Laplaciano de Gaussianas (LoG)/Diferencia de Gaussianas (DoG) para identificar posibles puntos de interés
  - Refinamiento de los puntos de interés para precisar su posición mediante el ajuste de un modelo continuo.
  - Asignación de una orientación (orientación dominante) en cada punto de interés.
  - Cómputo de un vector de 128 valores descriptores de las características de cada punto (*feature descriptor vector*) en función del histograma del gradiente local.

Espacio de escalas

- Filtrado con operador Laplaciano de Gaussianas aumentando la varianza.



Lindberg (1988), *Feature Detection with Automatic Scale Selection*

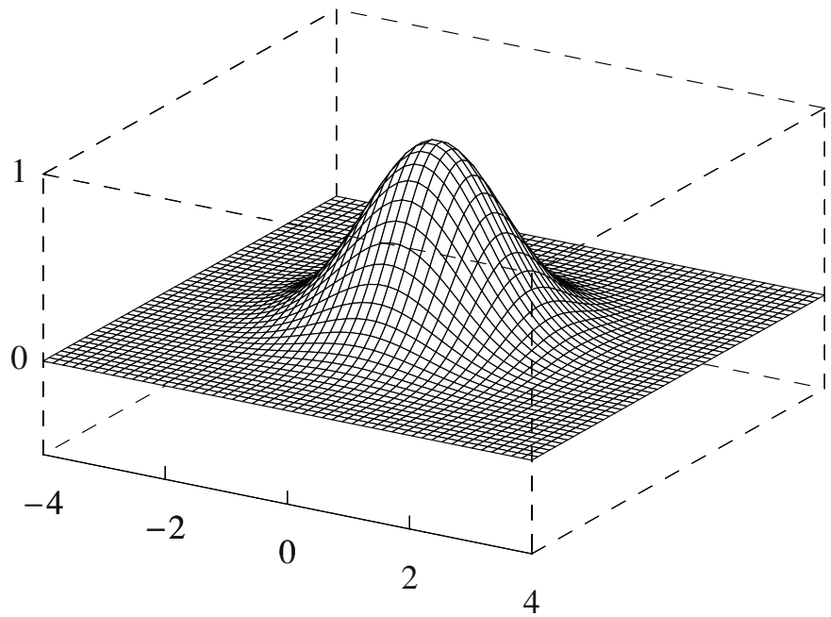


Laplaciano $\Delta f = \nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$

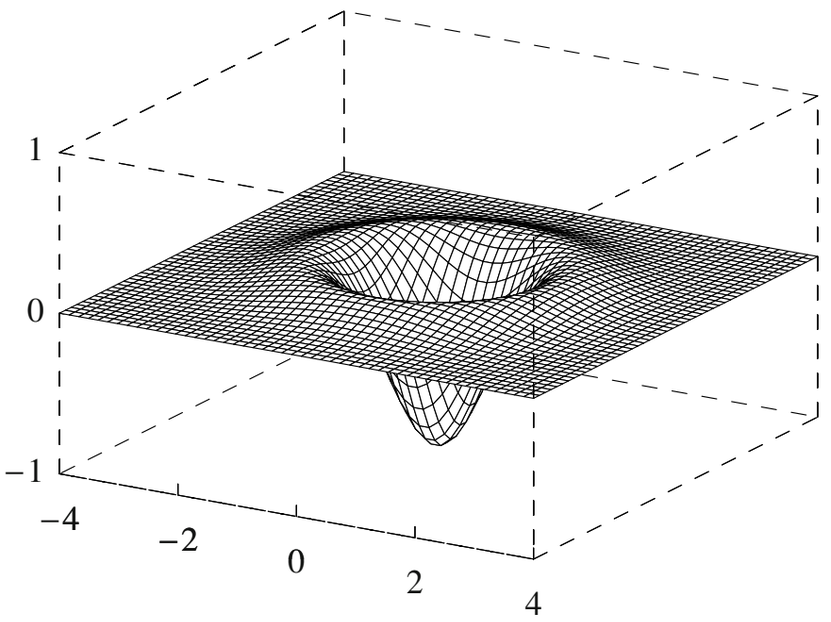
$f \in C^2$

Espacio de escalas

- Filtrado con operador Laplaciano de Gaussianas aumentando la varianza.



$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}}$$



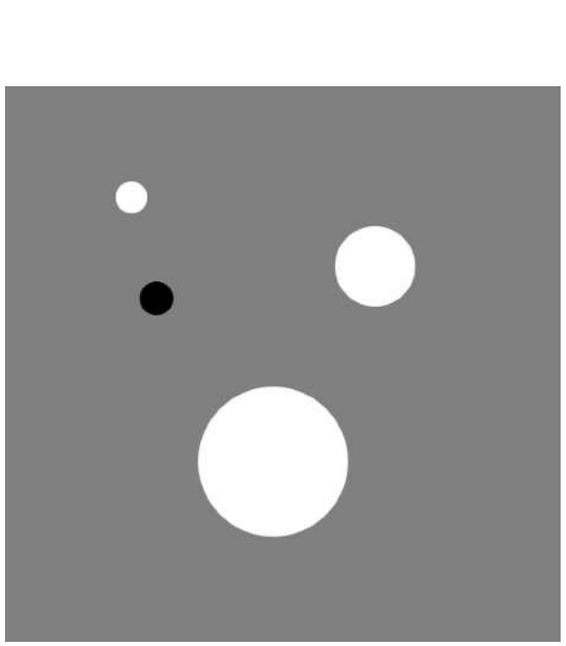
$$L_\sigma(x,y) = \nabla^2 G_\sigma(x,y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$$

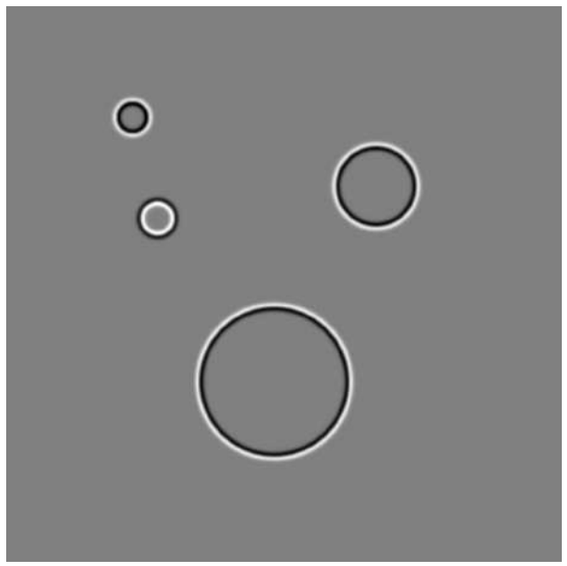
$$= \frac{1}{\pi\sigma^4}\left(\frac{x^2+y^2-2\sigma^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Respuesta máxima en puntos oscuros rodeados de un anillo claro de radio cercano a σ.

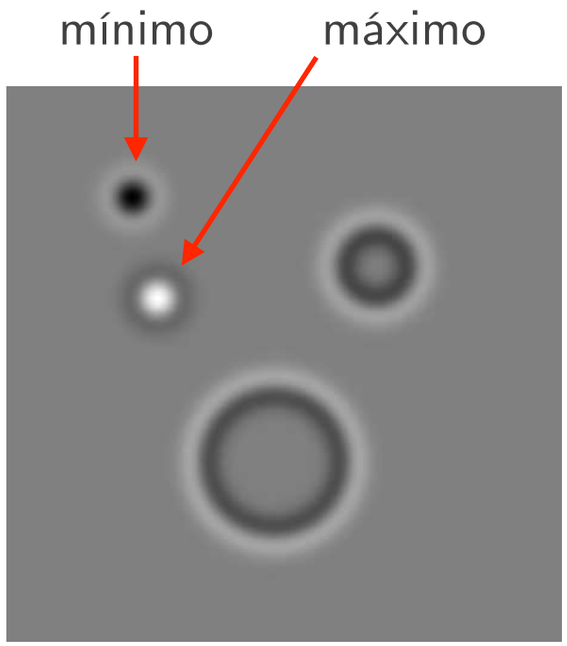
Espacio de escalas

- Filtrado con operador Laplaciano de Gaussianas aumentando la varianza.



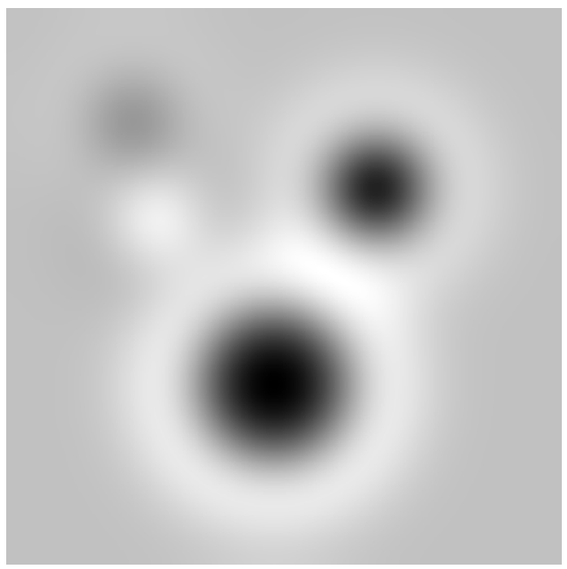


$\sigma = 2$

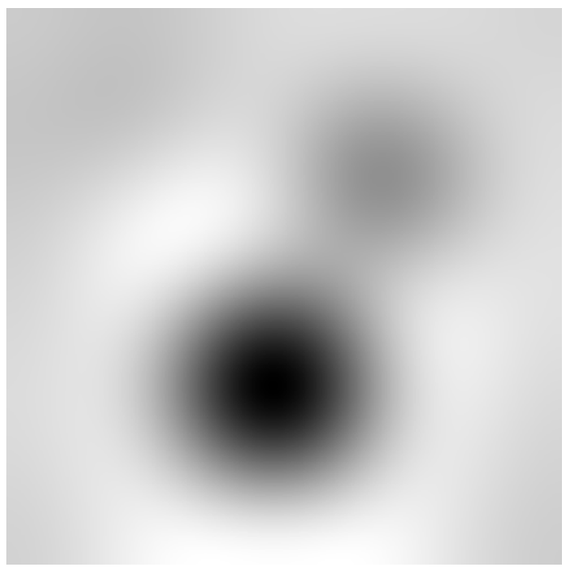


$\sigma = 10$

$\sigma = 20$



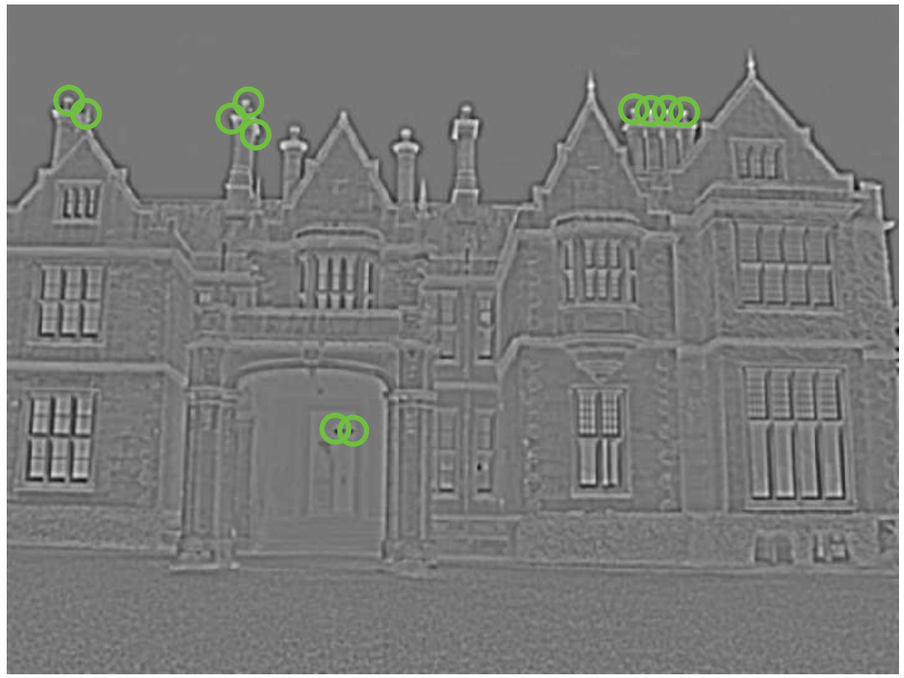
$\sigma = 40$



$\sigma = 80$

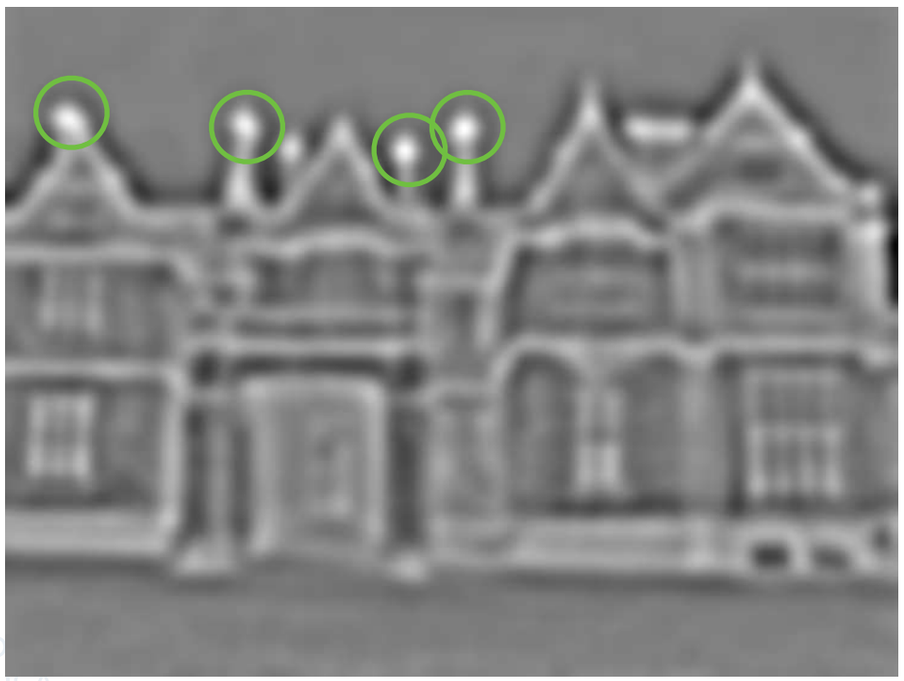
Espacio de escalas

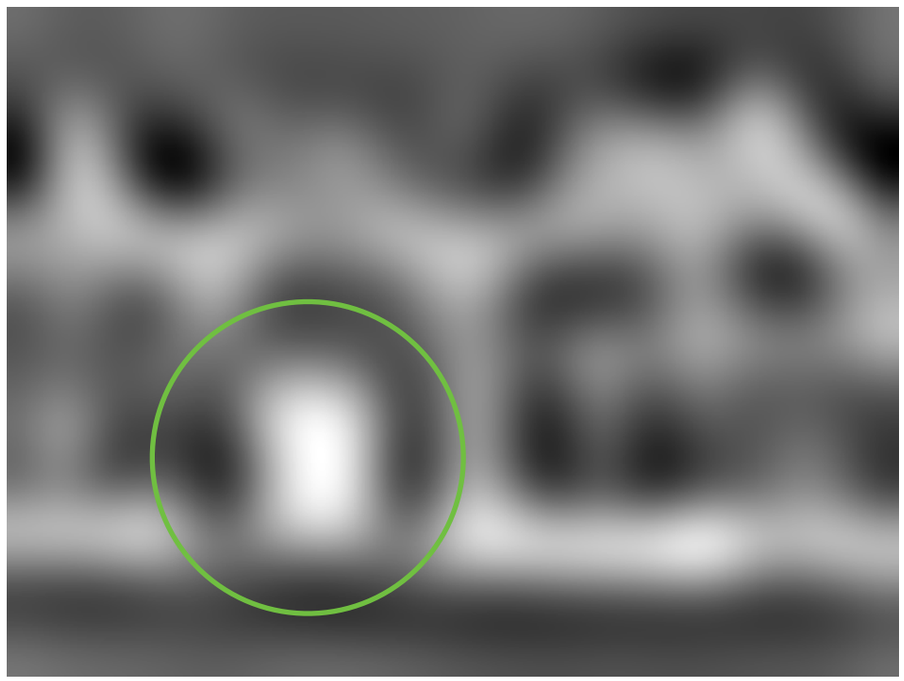
- Filtrado con operador Laplaciano de Gaussianas aumentando la varianza.



$\sigma = 2$

$\sigma = 10$

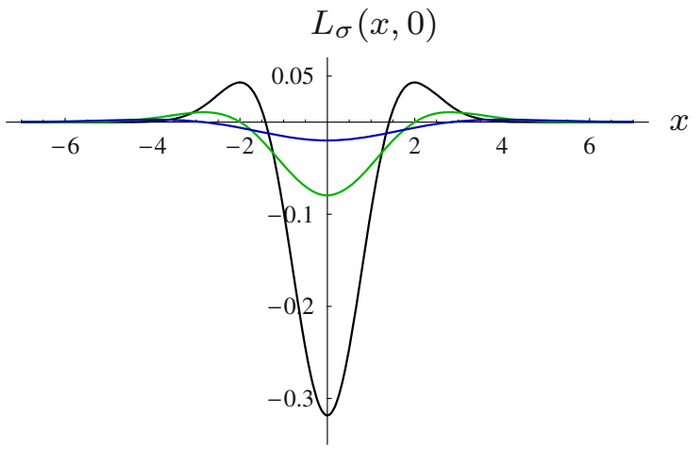


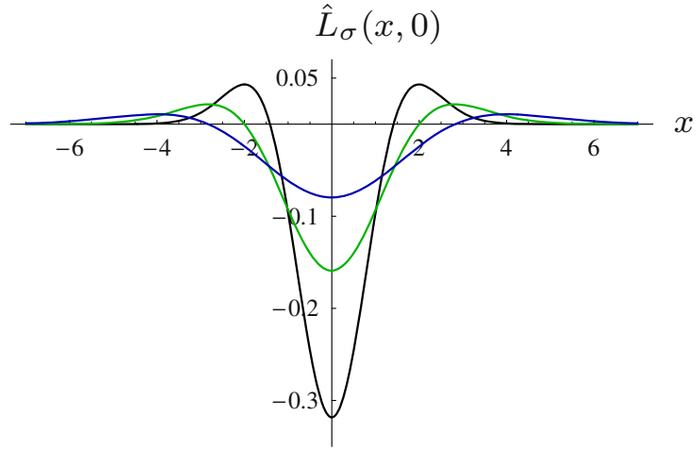


$\sigma = 40$

Laplacian of Gaussians (LoG) y Difference of Gaussians (DoG)

$$\hat{L}_\sigma(x,y) = \sigma^2 L_\sigma(x,y)$$

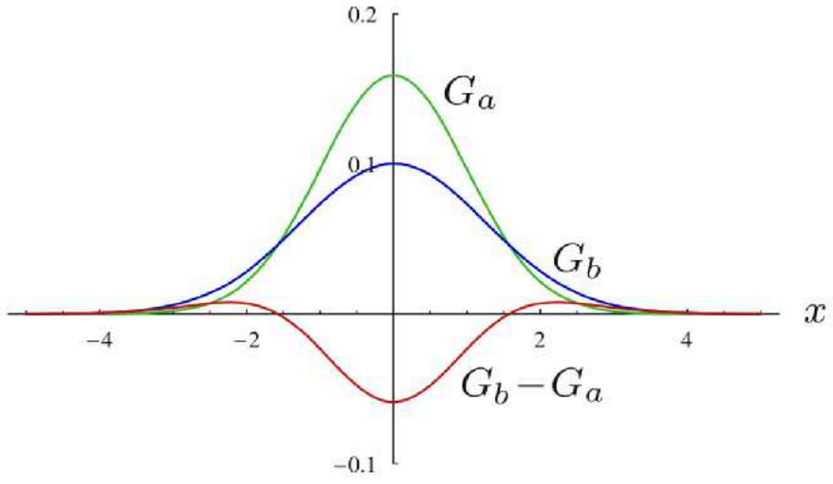




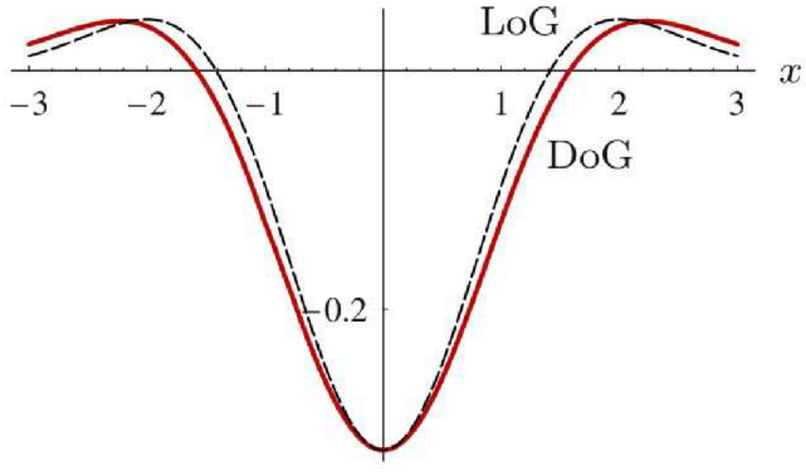
$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left|\hat{L}_\sigma(x,y)\right| \mathrm{d}x\,\mathrm{d}y = \frac{4}{e}$$

Independiente de la escala

- En la práctica se aproxima el LoG por la Diferencia de Gaussianas (DoG) de varianzas cercanas, con menor tiempo de cómputo.



$$\sigma_a = 1$$
$$\sigma_b = \kappa\,\sigma_a$$



$$\mathrm{DoG}_{\sigma,\kappa}(x,y) = G_b(x,y) - G_a(x,y)$$
$$\hat{L}_\sigma(x,y) = \hat{\kappa}\,\mathrm{DoG}_{\sigma,\kappa}(x,y)$$

Construcción del espacio de escalas

- Los puntos de interés se buscan como extremos locales en el espacio de DoG sobre múltiples escalas.
- Se construye un *espacio de escalas* como una sucesión de la aplicación del operador LoG/ DoG sobre la imagen de entrada.

$$(I * G_{\sigma_1}) * G_{\sigma_2} = I * G_{\sigma_{1,2}} \quad \text{donde} \quad \sigma_{1,2} = \sqrt{{\sigma_1}^2 + {\sigma_2}^2}$$

- Un *espacio de escalas Gaussiano* es un *vector* de imágenes compuesto por versiones de la imagen de entrada convolucionado con gaussianas de varianza creciente.

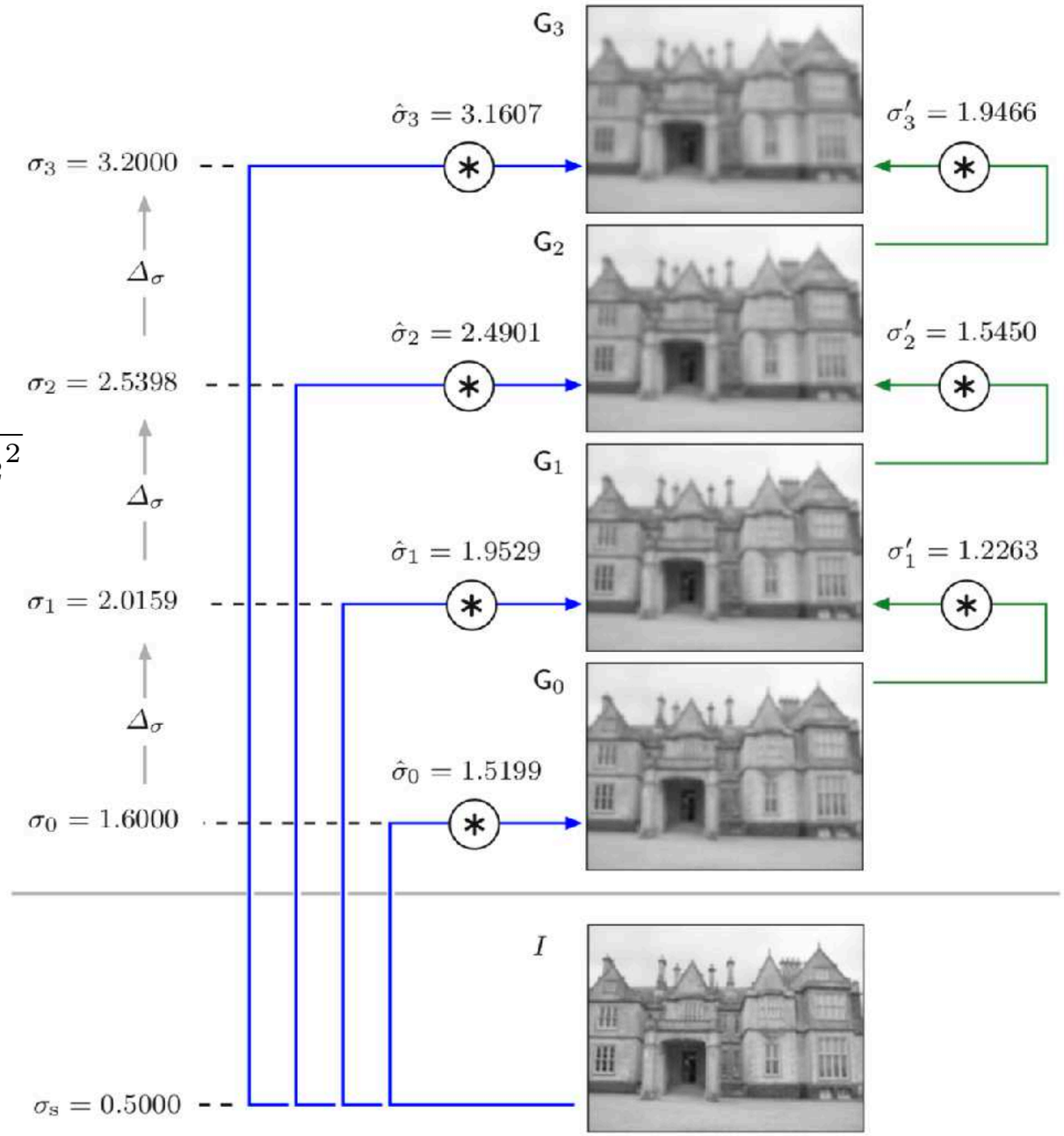
$$\mathsf{G} = (\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{M-1}) \quad \text{donde} \quad \mathsf{G}_m = I * G_{\sigma_m}$$

- En la implementación la relación entre dos escalas consecutivas viene dada por

$$\Delta_\sigma = \frac{\sigma_{m+1}}{\sigma_m} = 2^{1/Q}$$

  - Q define una *octava* (cada Q escalas se duplica).

$$\sigma_m = \sigma_0 \, \Delta_\sigma^m = \sigma_0 \, 2^{1/Q}$$



$$\hat{\sigma}_m = \sqrt{\sigma_m^2 - \sigma_s^2} = \sqrt{\sigma_0^2 \cdot 2^{2m/Q} - \sigma_s^2}$$

$$\sigma'_m = \sqrt{\sigma_m^2 - \sigma_{m-1}^2} = \sigma_0 \cdot 2^{m/Q} \cdot \sqrt{1 - 1/\Delta_\sigma^2}$$

(Detalles muy cercanos a la implementación)

Construcción del espacio de escalas

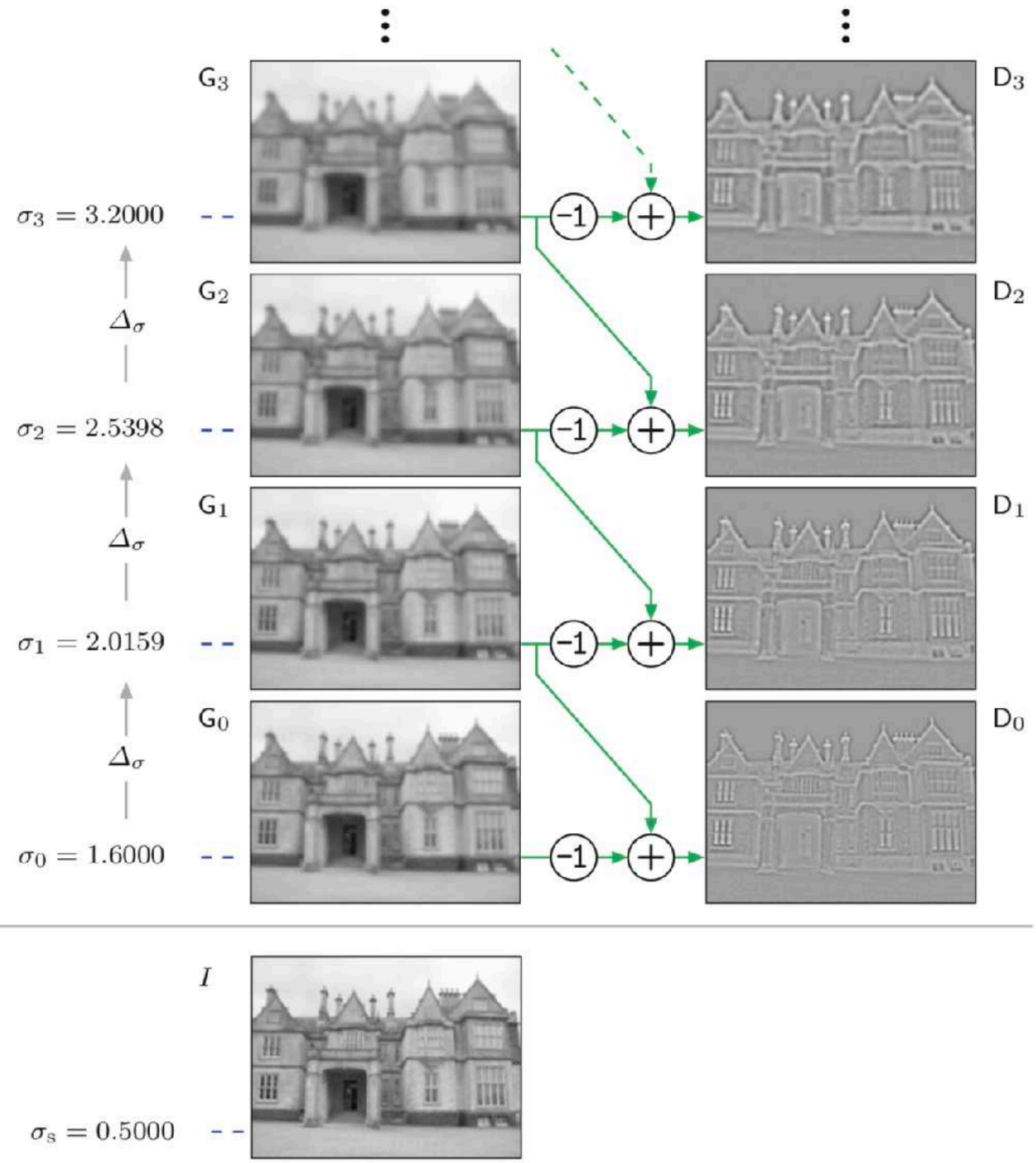
- A partir del *espacio de escalas Gaussiano* se construye el *espacio de escalas Laplaciano*

$$\mathsf{L} = (\mathsf{L}_0, \mathsf{L}_1, \ldots, \mathsf{L}_{M-1})$$

- Y su aproximación de DoG

$$\mathsf{D} = (\mathsf{D}_0, \mathsf{D}_1, \ldots, \mathsf{D}_{M-1})$$

$$\mathsf{D}_m = \hat{\kappa}\,(\mathsf{G}_{m+1} - \mathsf{G}_m) \approx \mathsf{L}_m$$



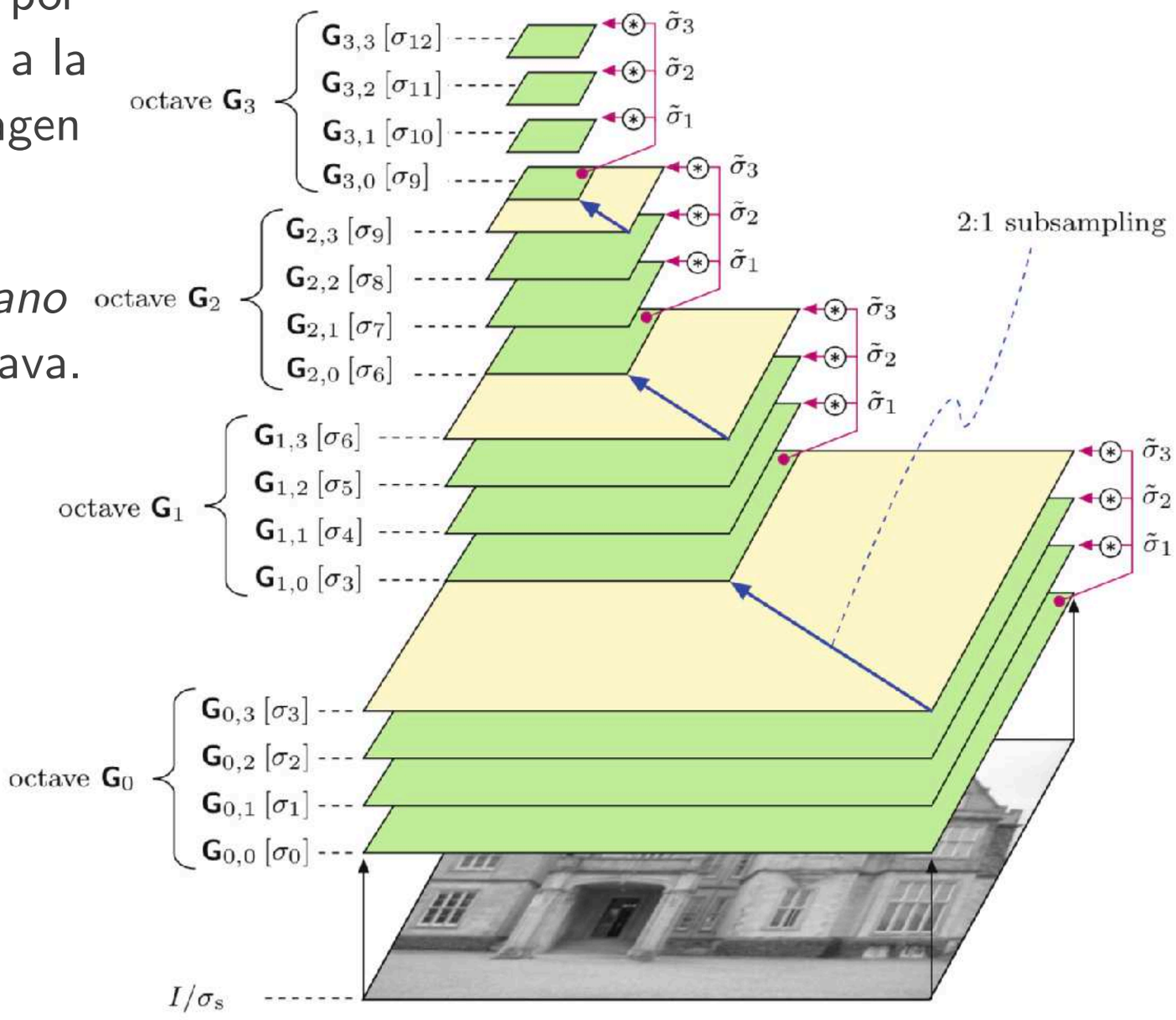
Construcción del espacio de escalas

- Cada Q escalas se duplica la escala por lo que se reduce el ancho de banda a la mitad, y puede submuestrear la imagen sin perder información.
- Se define *espacio de escalas Gaussiano jerárquico* con Q+1 escalas por octava.

$$\mathsf{G}_p = (\mathsf{G}_{p,0}, \mathsf{G}_{p,1}, \ldots, \mathsf{G}_{p,Q})$$

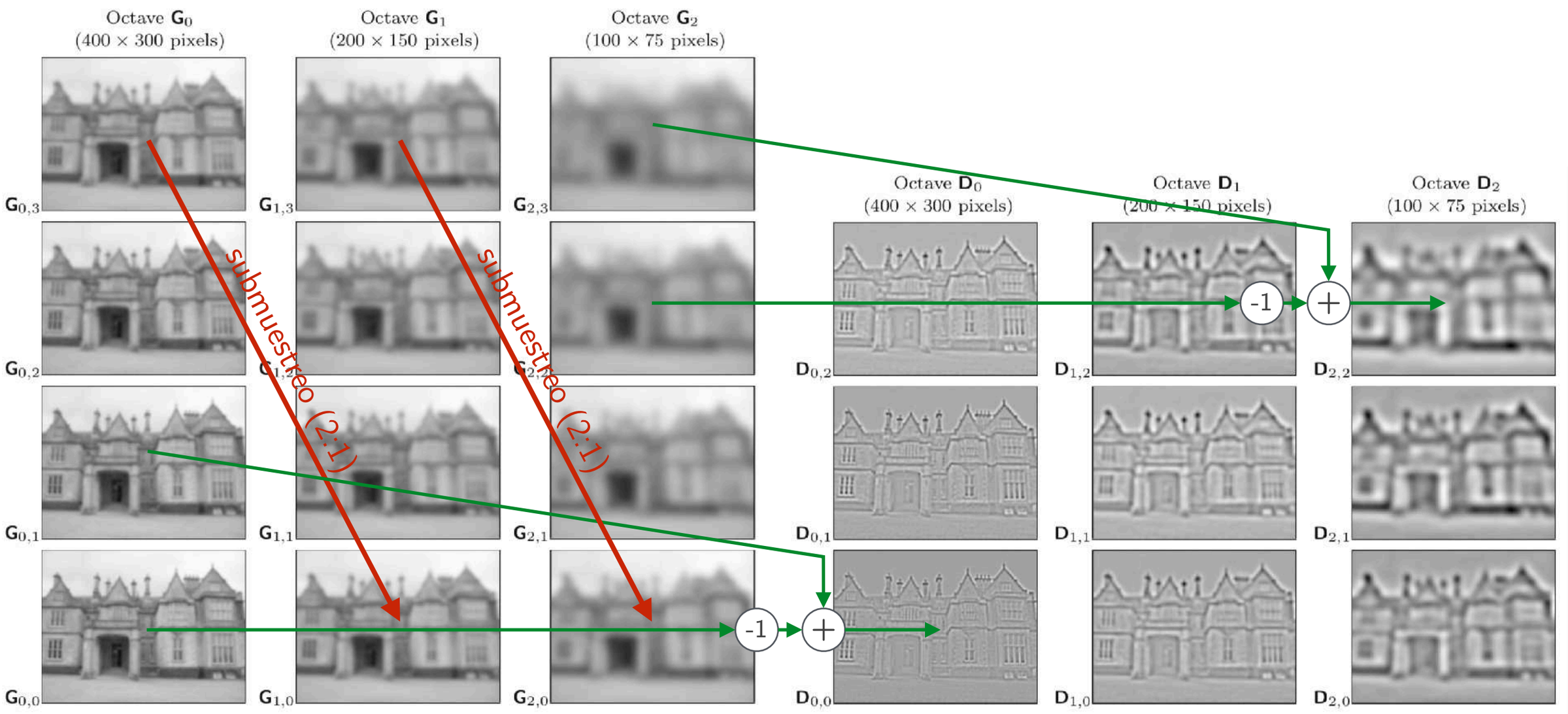
$$m = Qp + q$$

$$\mathsf{D}_{p,q} = \mathsf{G}_{p,q+1} - \mathsf{G}_{p,q}$$



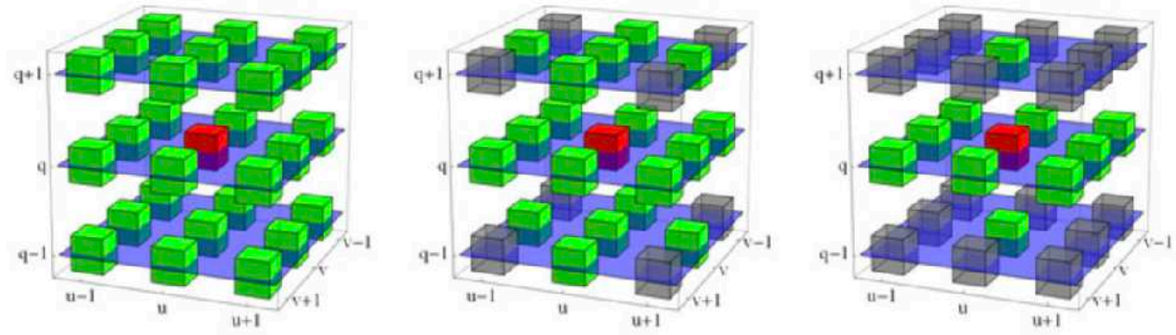
$$Q = 3$$

Construcción del espacio de escalas



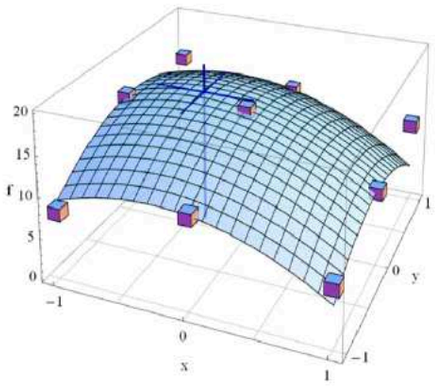
Selección y refinamiento de los puntos

- El proceso sigue con:
  - Búsqueda de extremos en el espacio de escalas DoG 3D.
    - La búsqueda se hace de forma independiente en cada escala.



(a) 26-neighborhood (b) 18-neighborhood (c) 10-neighborhood

Vecindarios de búsqueda de extremos.



- Refinamiento de la posición (interpolación subpixel).
  - Ajuste de una función cuadrática en el vecindario.

- Eliminación de puntos en bordes

$$H_{x,y}(c) = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{xy} & d_{yy} \end{bmatrix} \qquad \lambda_1 + \lambda_2 = \mathrm{tr}(H_{x,y}(c)) = d_{xx} + d_{yy}$$

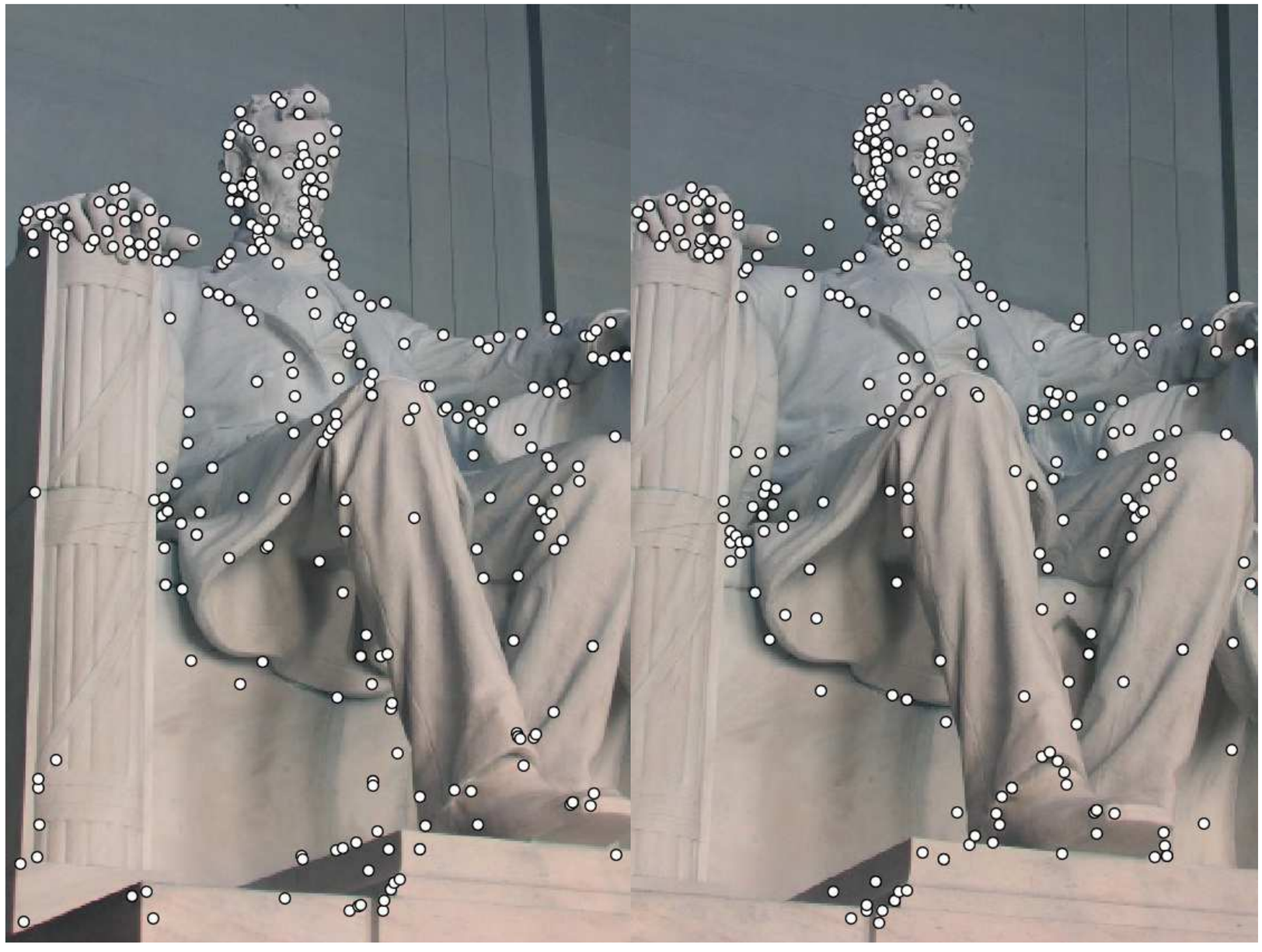
$$\lambda_1 \lambda_2 = \det(H_{x,y}(c)) = d_{xx} d_{yy} - 2 d_{xy}^2 \qquad \rho_{1,2} = \frac{\lambda_1}{\lambda_2}$$

$$\alpha = \frac{\mathrm{tr}(H_{x,y}(c))^2}{\det(H_{x,y}(c)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(\rho_{1,2} + 1)^2}{\rho_{1,2}} \leq \alpha_{\max} = \frac{(\bar{\rho}_{1,2} + 1)^2}{\bar{\rho}_{1,2}}$$

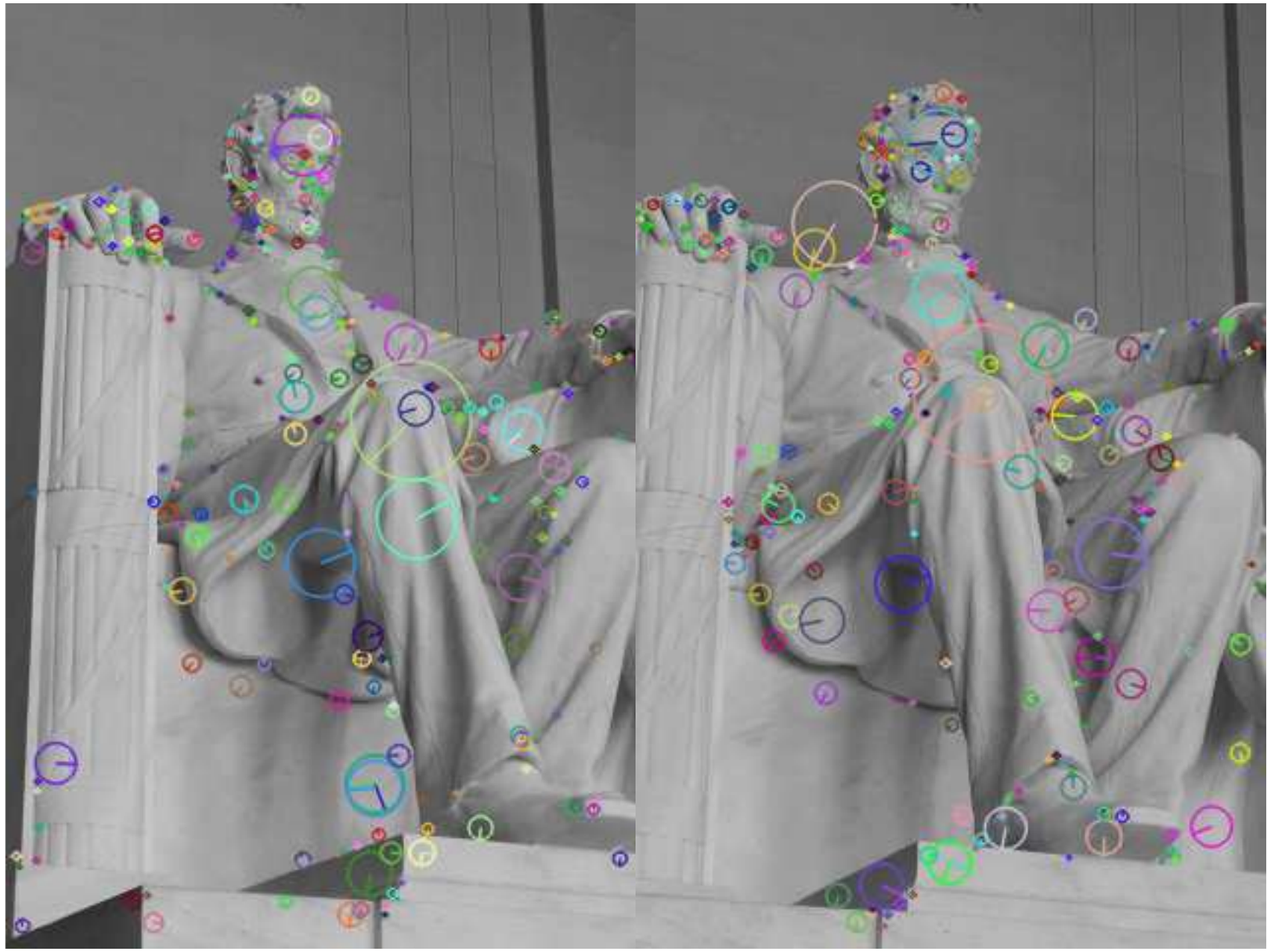
SIFT

SIFT

SIFT



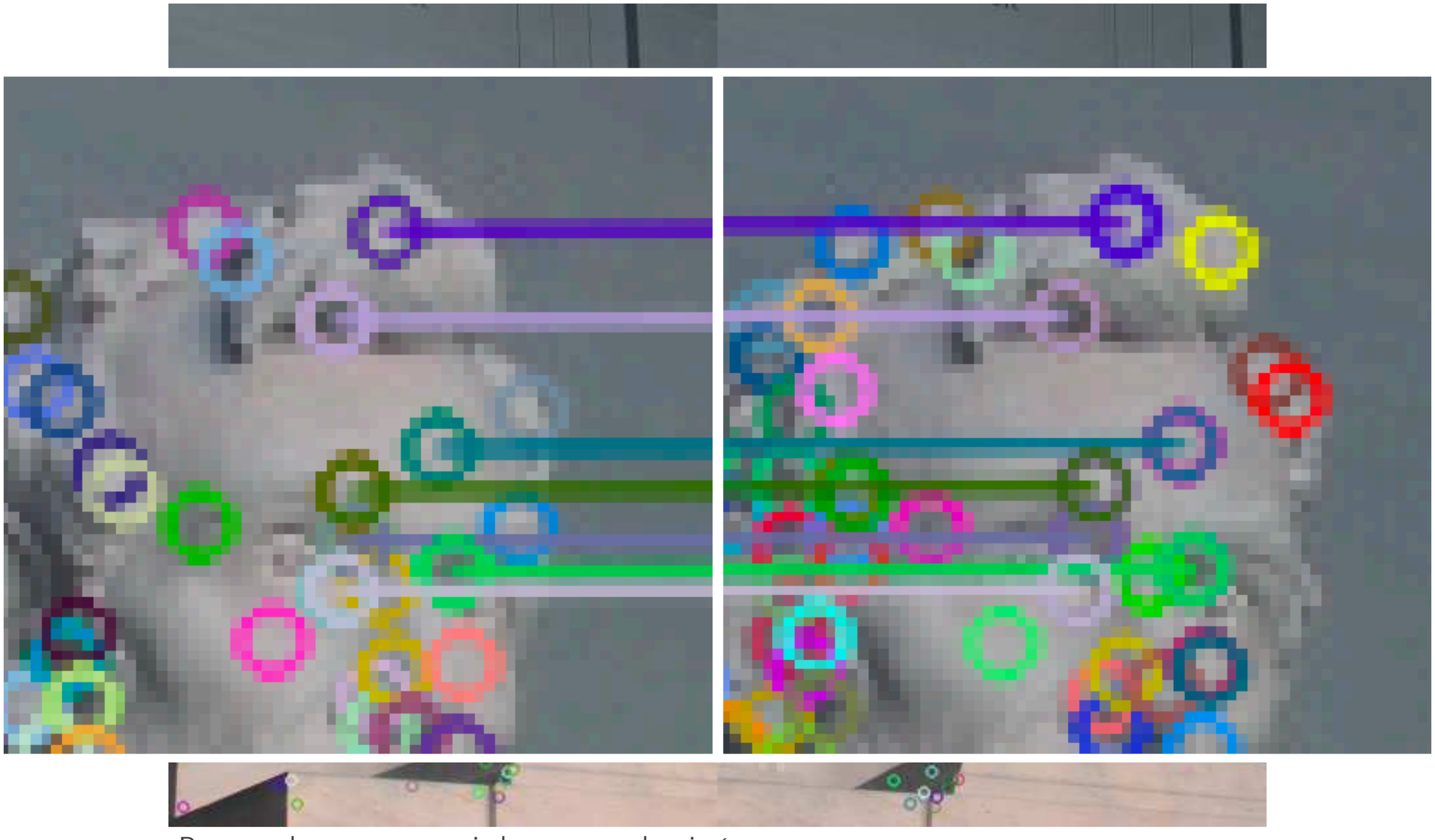
SIFT



Codificación de la representación de los puntos: cada círculo centrado en un punto relevante, con radio dado por la escala donde se encontró el punto, la dirección marcada es la dirección principal del *vector de características*.

¿vector de características?

SIFT



Puntos relevantes emparejados entre ambas imágenes.

Vector de características de SIFT

- El procedimiento anterior deja un lista de puntos candidatos (*keypoints*) $\mathbf{k}(p,q,u,v)$
- Para cada uno de estos puntos se calculan hasta cuatro vectores de características o descriptores locales.
  - Primero, en cada punto, se buscan las orientaciones dominantes (hasta cuatro) de la distribución de los gradientes en el espacio de escala correspondiente.
  - Para cada orientación se crea un vector de 128 dimensiones dadas por magnitudes de gradientes (8 direcciones en una grilla de 16 puntos)
  - Estos gradientes son pre-calculados en escalas relevantes del espacio de escalas.

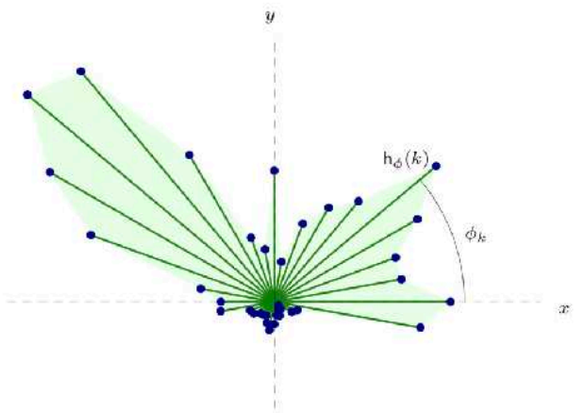
$$\nabla_{p,q}(u,v) = \begin{bmatrix} A_{p,q}(u,v) \\ B_{p,q}(u,v) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \mathsf{G}_{p,q}(u+1,v) - \mathsf{G}_{p,q}(u-1,v) \\ \mathsf{G}_{p,q}(u,v+1) - \mathsf{G}_{p,q}(u,v-1) \end{bmatrix}$$

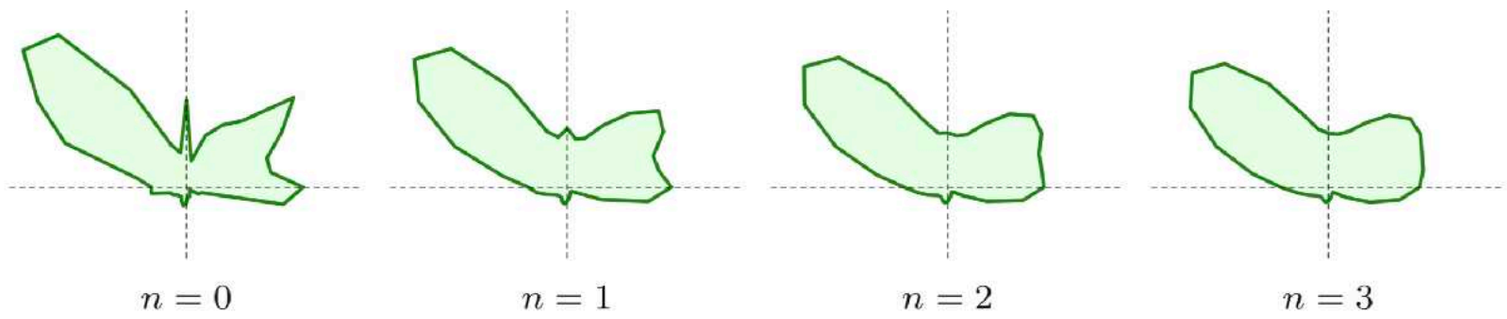
$$m_{p,q}(u,v) = \|\nabla_{p,q}(u,v)\| = \sqrt{A^2_{p,q}(u,v) + B^2_{p,q}(u,v)}$$

$$\phi_{p,q}(u,v) = \angle\nabla_{p,q}(u,v) = \tan^{-1}\left(\frac{B^2_{p,q}(u,v)}{A^2_{p,q}(u,v)}\right)$$

Vector de características de SIFT

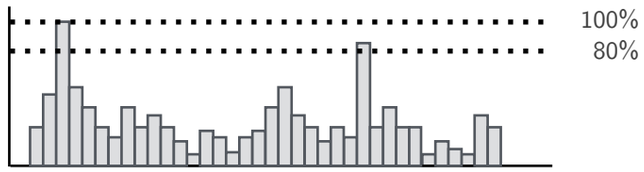
- La *orientación dominante* se determina a partir de un histograma de gradientes en 36 orientaciones centradas en el punto (10º de resolución).





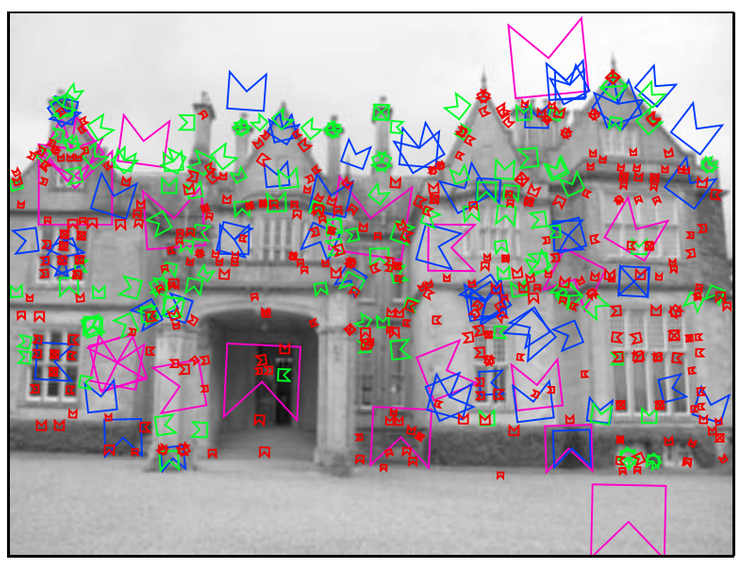
Suavizado circular del histograma.

- Una orientación es considerada significante si es un máximo local y su magnitud no es menor a una fracción (0.8) del máximo global.



- Definir una orientación dominante permite agregar invarianza a rotaciones.

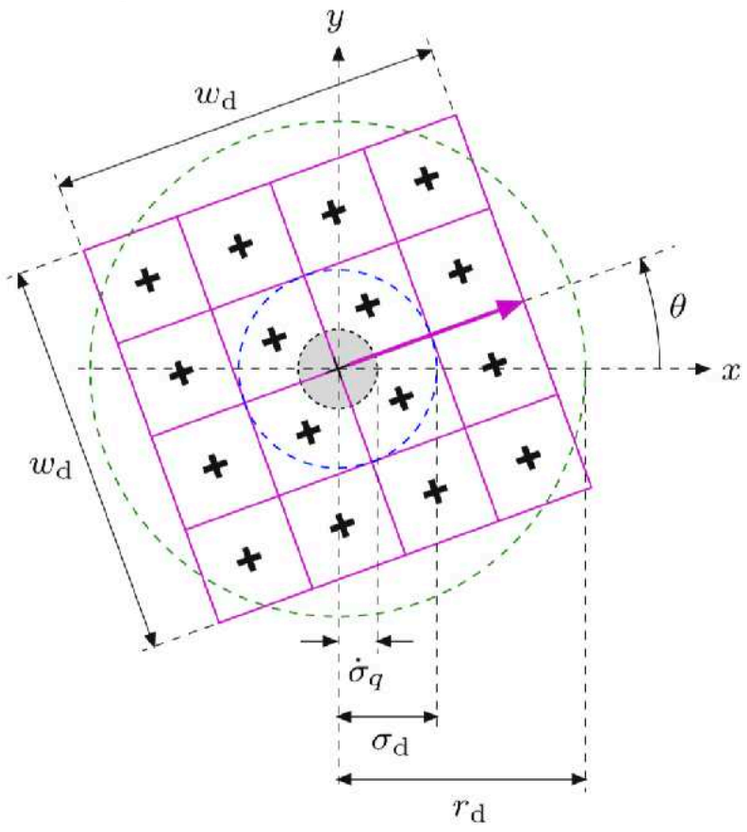
Histogramas suavizados.

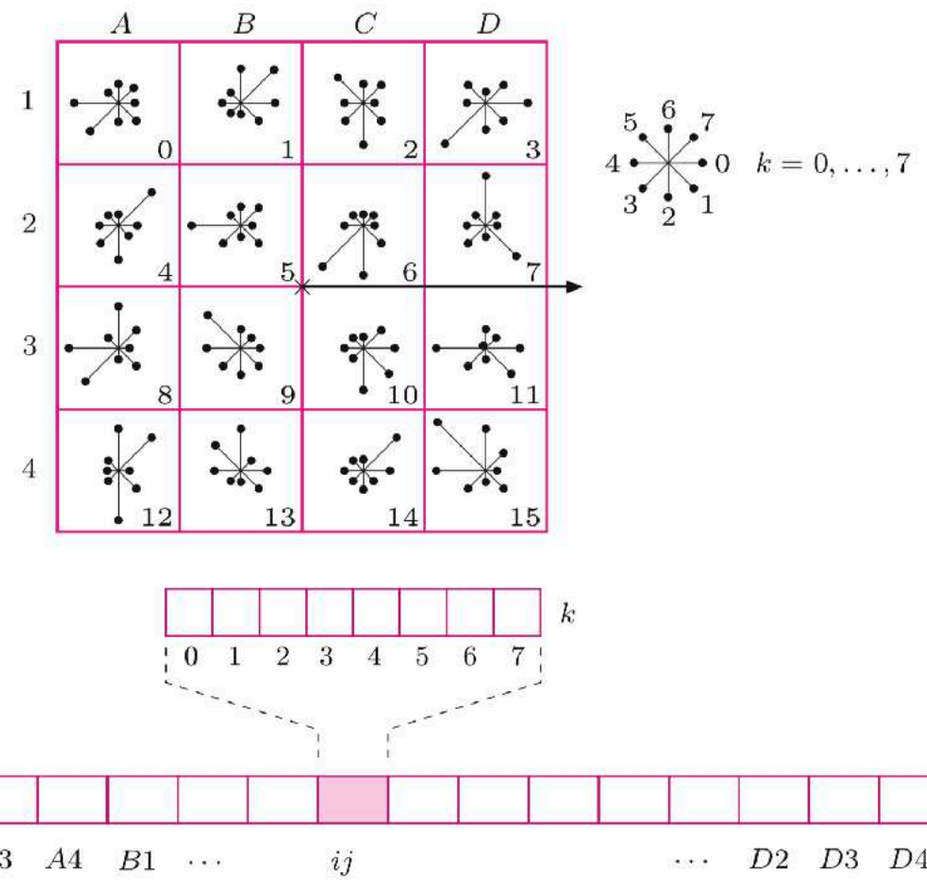


Orientaciones dominantes.

Vector de características de SIFT

- Para cada punto $\mathbf{k}(p,q,u,v)$ y su dirección dominante θ, el correspondiente descriptor SIFT se obtiene muestreando los gradientes en la octava p y nivel q del espacio de escala Gaussiano G.
- El descriptor es calculado a partir de una región soporte cuadrada centrada en el punto y alineada con su dirección dominante y partida en (4 × 4) sub-cuadrados.





8 x 16 gradientes determinan el vector de características $\boldsymbol{f}_{\mathrm{SIFT}}$ de 128 dimensiones

Vector de características de SIFT

- La definición de las parejas se basa en una métrica de similitud (L1, L2, ...) entre los puntos detectados $\boldsymbol{s}_k^{(j)}$ que incluyen los vectores de características de I_j.

$$S^{(j)} = (\boldsymbol{s}_1^{(j)}, \boldsymbol{s}_2^{(j)}, \ldots, \boldsymbol{s}_{N_j}^{(j)}) \qquad \boldsymbol{s}_i = (x_i, y_i, \sigma_i, \theta_i, \boldsymbol{f}_i)$$

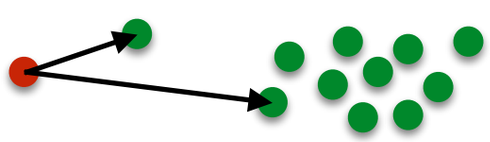
$$\mathrm{dist}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \|\boldsymbol{f}_i - \boldsymbol{f}_j\|$$

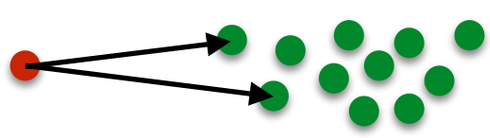
- Parejas de puntos correspondientes deben tener un distancia pequeña. Sin embargo, umbralizar respecto a un valor fijo puede dejar afuera parejas correctas. Se usa el cociente de distancia a los dos primeros correspondientes.

$$\boldsymbol{s}_1 = \arg \min_{\boldsymbol{s}_j \in S^{(b)}} \mathrm{dist}(\boldsymbol{s}_r, \boldsymbol{s}_j), \boldsymbol{s}_r \in S^{(a)}$$

$$\boldsymbol{s}_2 = \arg \min_{\substack{\boldsymbol{s}_j \in S^{(b)} \\ \boldsymbol{s}_2 \neq \boldsymbol{s}_1}} \mathrm{dist}(\boldsymbol{s}_r, \boldsymbol{s}_j), \boldsymbol{s}_r \in S^{(a)}$$

$$\rho_{\mathrm{match}}(\boldsymbol{s}_r, \boldsymbol{s}_1, \boldsymbol{s}_2) = \frac{\mathrm{dist}(\boldsymbol{s}_r, \boldsymbol{s}_1)}{\mathrm{dist}(\boldsymbol{s}_r, \boldsymbol{s}_2)} \leq \bar{\rho}_{\mathrm{match}}$$





Speeded-Up Robust Features (SURF)

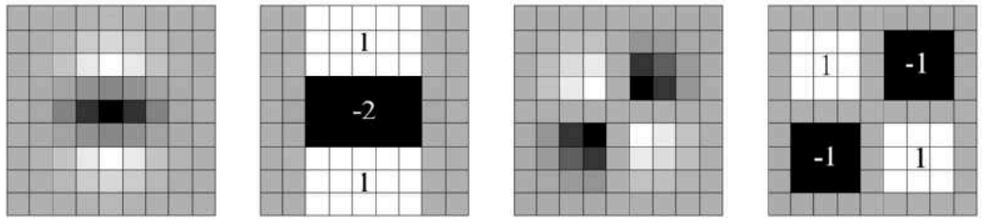
- SURF simplifica varias etapas del cómputo para acelerar la búsqueda.
- Aproxima la Hessiana para la detección de puntos.
- Se basa en *integral images* para la aproximación de convoluciones (*Fast-Hessian*)
- Puede paralelizarse para diferentes escalas en el *espacio de escalas*.
- Usa (Haar) wavelets en la horizontal y la vertical para extraer características en un vecindario.
- Vector de características de 64 (128) dimensiones con similar robustez a SIFT.

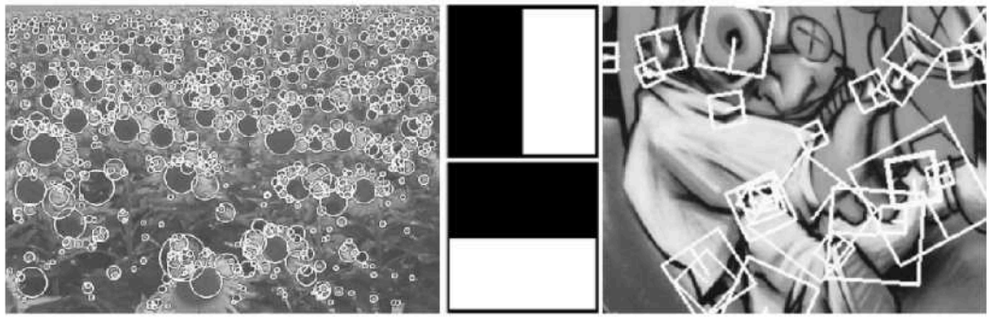
SURF: Speeded Up Robust Features

Herbert Bay[1], Tinne Tuytelaars[2], and Luc Van Gool[12]

[1] ETH Zurich
{bay, vangool}@vision.ee.ethz.ch
[2] Katholieke Universiteit Leuven
{Tinne.Tuytelaars, Luc.Vangool}@esat.kuleuven.be

Abstract. In this paper, we present a novel scale- and rotation-invariant interest point detector and descriptor, coined SURF (Speeded Up Robust Features). It approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster.
This is achieved by relying on integral images for image convolutions; by building on the strengths of the leading existing detectors and descriptors (*in casu*, using a Hessian matrix-based measure for the detector, and a distribution-based descriptor); and by simplifying these methods to the essential. This leads to a combination of novel detection, description, and matching steps. The paper presents experimental results on a standard evaluation set, as well as on imagery obtained in the context of a real-life object recognition application. Both show SURF's strong performance.





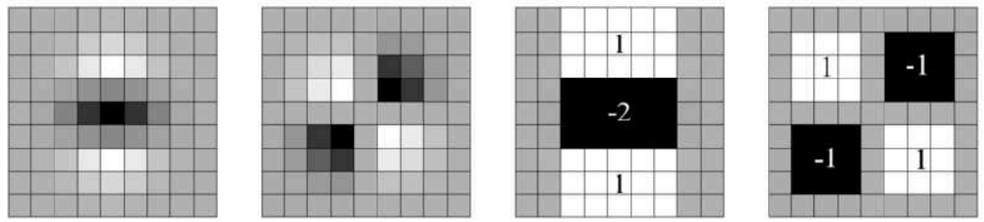
Aproximaciones en el detector de SURF

- Fast-Hessian

Derivada segunda de la Gaussiana convolucionada con la imagen.

$$\mathcal{H}(\mathbf{x},\sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{bmatrix}$$

Aproximaciones en filtros 9×9 con σ=1.2.



$L_{yy}(\mathbf{x},\sigma)$ $L_{xy}(\mathbf{x},\sigma)$ $D_{yy}(\mathbf{x},\sigma)$ $D_{xy}(\mathbf{x},\sigma)$

$$\det(\mathcal{H}_{\text{aprox}}) = D_{xx}D_{yy} - 0.81D_{xy}^2$$

- *Integral images*

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

$I(x,y)$

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

$$I_\Sigma(x,y) = \sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i,j)$$

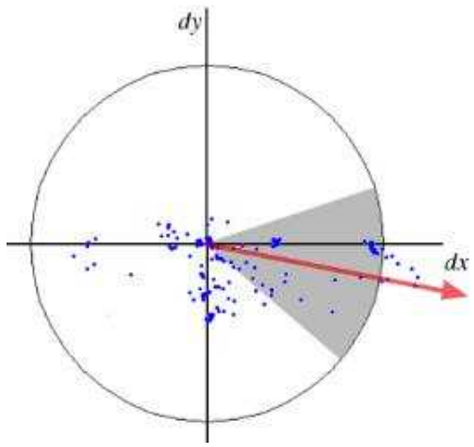
$$\sum_{\substack{x_0<x\le x_1 \\ y_0<y\le y_1}} I(x,y) = \begin{matrix} I_\Sigma(x_1,y_1) + I_\Sigma(x_0,y_0) \\ -I_\Sigma(x_1,y_0) - I_\Sigma(x_0,y_1) \end{matrix}$$

3+2+5+4 = 14 = 46+10-22-20

- Espacio de escalas: aumento del tamaño del filtro (9×9, 15×15, 21×21, 27×27) y del σ correspondiente sin necesidad de submuestreo.
- Supresión de máximos en vecindario 3×3×3 con interpolación en la imagen y escalas.

Descriptor de SURF

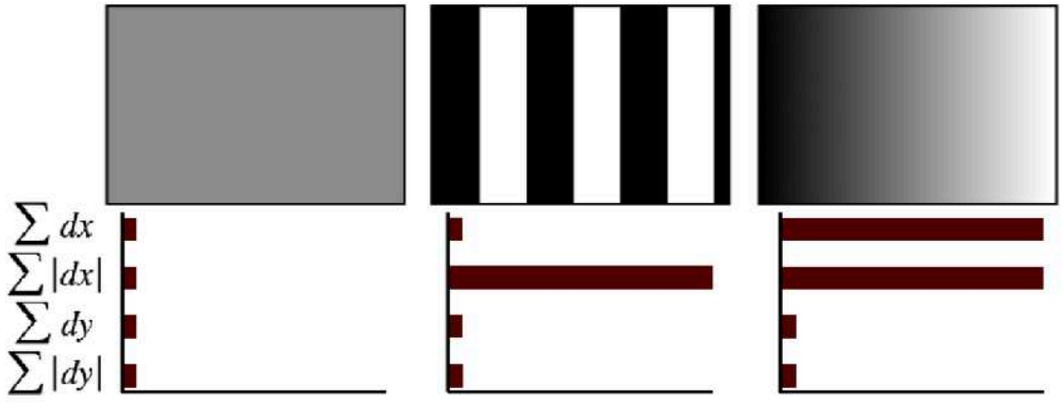
- Respuesta a wavelet de Haar en vertical y horizontal en un vecindario circular de radio $6\sigma_k$. La orientación dominante se estima a partir del vector de máximo módulo creado con el aporte de todas las orientaciones en una ventana deslizante de 60º. Esto no se hace en la variante U-SURF (más rápida aún).



Los cuadrados muestran la posición, escala y orientación del punto detectado.

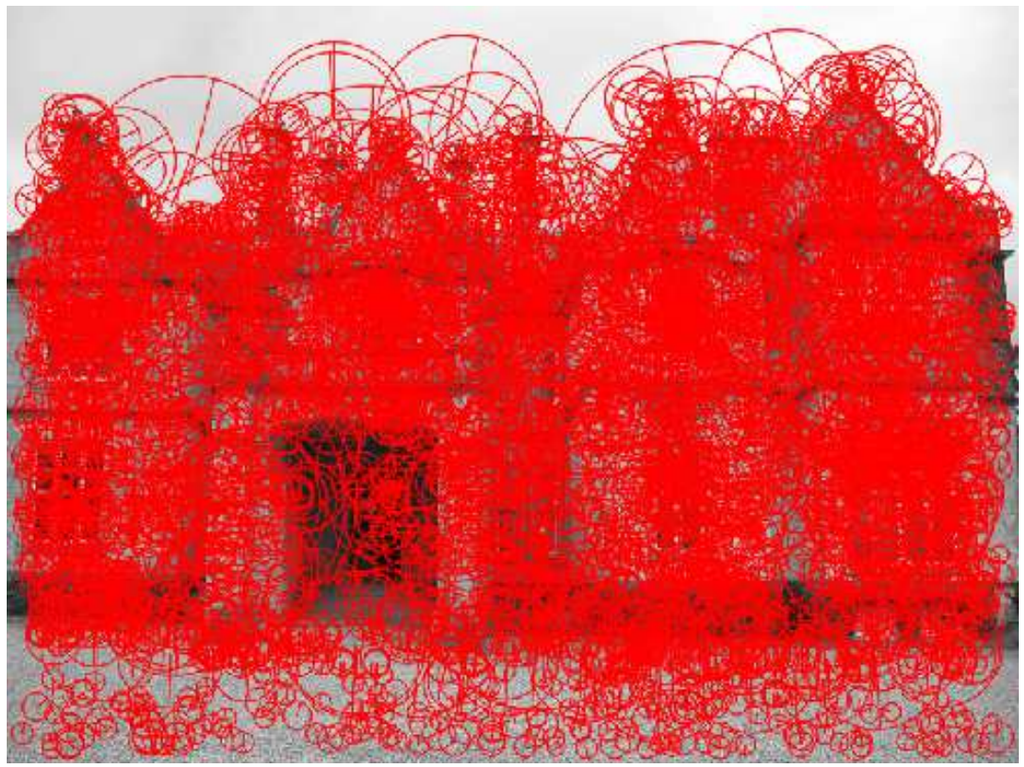
- Una región cuadrada de $20\sigma_k$ de lado se divide en 4×4 donde se calculan las derivadas horizontales y verticales; con ellas se forma un vector de características en cada uno de las 16 sub-regiones

$$\mathbf{v}_i = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right)$$



Tres ejemplo de los valores que toma el descriptor $\mathbf{v}_i$.

Ejemplos de SURF



Umbral de Hessiano = 100

Umbral de Hessiano = 1000

Umbral de Hessiano = 5000

Umbral de Hessiano = 10000

Reconocimiento de objetos

- Tenemos procedimiento(s) para detectar puntos y extraer descriptores de sus características locales, ¿cómo lo usamos para reconocer un objeto en una nueva imagen?

- Es posible armar un detector a partir del matcheo de los descriptores a partir de una cierta métrica (L1, L2, ...) y tener en cuenta la distribución de los puntos.

Reconocimiento de objetos

SURF, H_{Th}=100

Reconocimiento de objetos

SURF, H_{Th}=1000

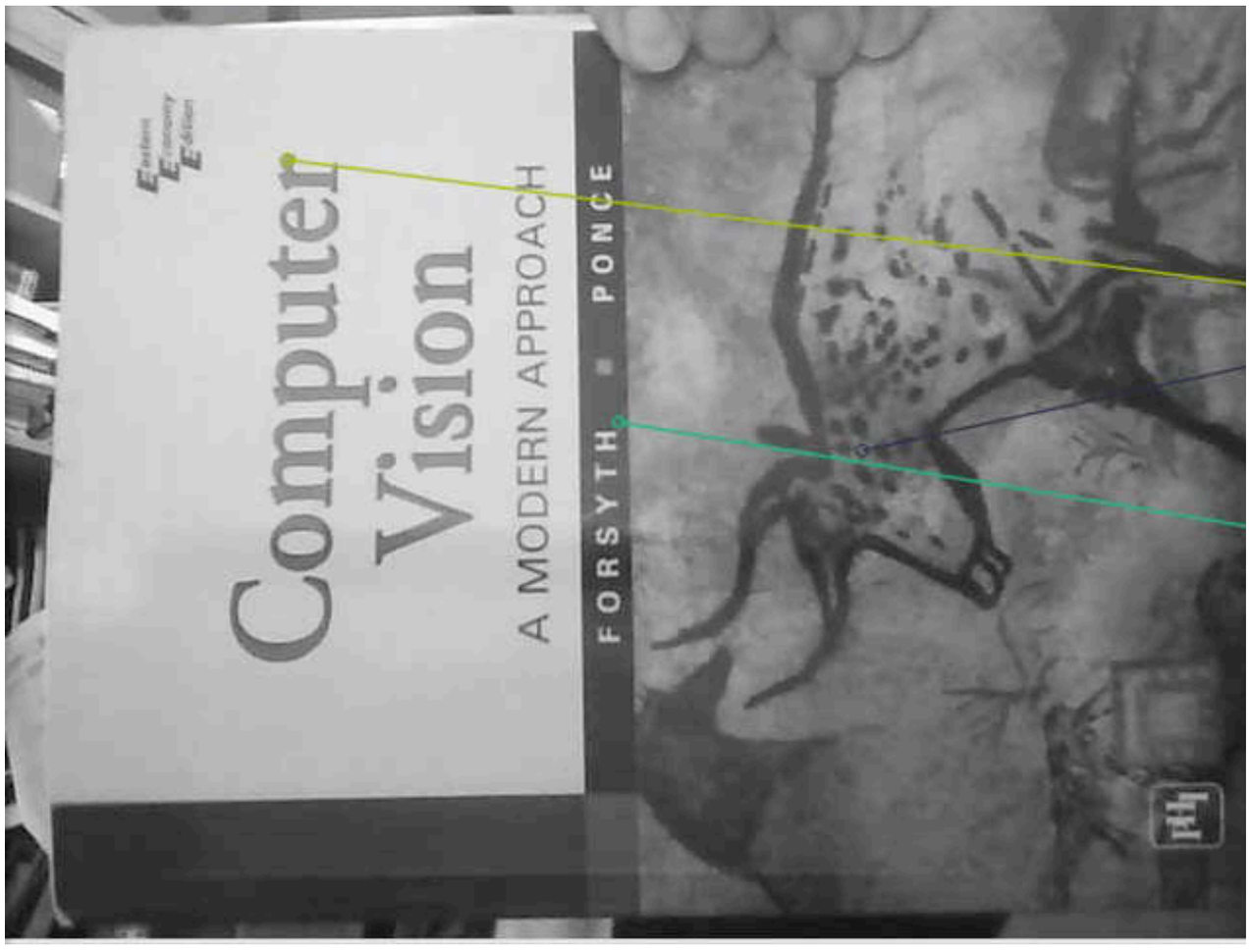
Reconocimiento de objetos

SURF, H_{Th}=100

Reconocimiento de objetos

SURF, H_{Th}=1000

Reconocimiento de objetos



Objeto a buscar el en video.

Video

Reconocimiento de objetos

SIFT, SURF, python y OpenCV

- Ejemplo SIFT

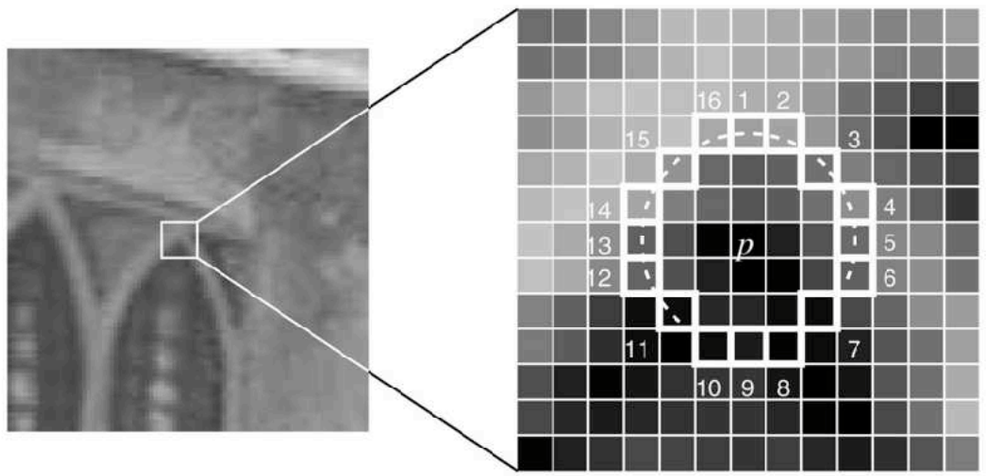
```
import cv2
import sys
from matplotlib import pyplot as plt

# Leer la imagen de los argumentos
filename = str(sys.argv[1])
img = cv2.imread(filename)

# Crear una instancia de un detector SIFT
sift = cv2.xfeatures2d.SIFT_create()

# Detectar puntos y calcular descriptores
kp, des = sift.detectAndCompute(img,None)

# Dibujar los puntos detectados
img2 = cv2.drawKeypoints(img,kp,None,(0,0,255),4)
plt.imshow(img2)
plt.title('SIFT'), plt.axis('off'), plt.show()
```

- Ejemplo SURF

```
import cv2
import sys
from matplotlib import pyplot as plt

# Leer la imagen de los argumentos
filename = str(sys.argv[1])
img = cv2.imread(filename)

# Crear una instancia de un detector SURF
Ht = 5000  # Umbral sobre la Hessiana
surf = cv2.xfeatures2d.SURF_create(Ht)

# Detectar puntos y calcular descriptores
# surf.setUpright(True)
kp, des = surf.detectAndCompute(img, None)

# Dibujar los puntos detectados
img2 = cv2.drawKeypoints(img, kp, None,\
    flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
plt.imshow(img2)
plt.title('SURF'), plt.axis('off'), plt.show()
```

Otros detectores y descriptores

- Features from Accelerated Segment Test (FAST, 2010)



n=12

  - Objetivo es la velocidad
  - Es un candidato si más de n pixeles consecutivos (en un arco) son más/menos brillante que el centro (I_p+t/I_p-t), donde t es un umbral.
    - Una verificación más rápida es con 1 y 9, 5 y 13, al menos tres.
  - Introduce conceptos de aprendizaje automático para el descriptor, basado en un clasificador con un *árbol de decisión*.
  - Poco robusto al ruido. Dependencia con el umbral.
  - Disponible en opencv: `cv2.FastFeatureDetector.detect`

Otros detectores y descriptores

- Binary Robust Independent Elementary Features (BRIEF)
  - Representación binaria de los descriptores, busca reducir la memoria usada (128 float = 512 bytes) para dispositivos de recursos limitados.
  - Cálculo de comparaciones más eficiente (Hamming es XOR y suma).
- La combinación de FAST y BRIEF se conoce como: Oriented FAST and Rotated BRIEF (ORB)

```python
import cv2

img = cv2.imread('imagenes/castle.tif')

# Iniciar detector STAR.
orb = cv2.ORB_create()

# Detectar y describir los keypoints con ORB.
kp, des = orb.detectAndCompute(img, None)

# Dibujar los keypoints.
img2 = cv2.drawKeypoints(img, kp, None, color=(0, 255, 0), flags=0)

# Mostrar la imagen.
cv2.imshow('ORB detector', img2)
```

Affine SIFT

IPOL
IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2011–02–24.
Submitted on 2011–00–00, accepted on 2011–00–00.
ISSN 2105–1232 © 2011 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2011.my-asift

ASIFT: An Algorithm for Fully Affine Invariant Comparison

Guoshen Yu[1], Jean-Michel Morel[2]

[1] CMAP, École Polytechnique, France (yu@cmap.polytechnique.fr)
[2] CMLA, ENS Cachan, France (moreljeanmichel@gmail.com)

Abstract

If a physical object has a smooth or piecewise smooth boundary, its images obtained by cameras in varying positions undergo smooth apparent deformations. These deformations are locally well approximated by affine transforms of the image plane. In consequence the solid object recognition problem has often been led back to the computation of affine invariant image local features. The similarity invariance (invariance to translation, rotation, and zoom) is dealt with rigorously by the SIFT method The method illustrated and demonstrated in this work, Affine-SIFT (ASIFT), simulates a set of sample views of the initial images, obtainable by varying the two camera axis orientation parameters, namely the latitude and the longitude angles, which are not treated by the SIFT method. Then it applies the SIFT method itself to all images thus generated. Thus, ASIFT covers effectively all six parameters of the affine transform.

Reconocimiento de objetos

Find-Object

Simple Qt interface to try OpenCV implementations of SIFT, SURF, FAST, BRIEF and other feature detectors and descriptors.

http://introlab.github.io/find-object/

Bibliografía

- Szeliski, R. *Computer Vision: Algorithms and Applications*. Springer, 2010 (http://szeliski.org/Book)
- Burger, W., Burge, M., *Principles of Digital Image Processing: Core Algorithms*. Springer, 2009.
- Burger, W., Burge, M., *Principles of Digital Image Processing: Advanced Methods*. Springer, 2013.
- Harris, C., & Stephens, M. (1988, August). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, No. 50, pp. 10-5244).
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91-110.
- Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg.
- Rosten, E., Porter, R., & Drummond, T. (2008). Faster and better: A machine learning approach to corner detection. IEEE transactions on pattern analysis and machine intelligence, 32(1), 105-119.