SIFT: Scale-Invariant Feature Transform

Agustín Ballesteros, María Delfina Kiss, Marcos Rostan y Julián Rugo

Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires

1 Introducción

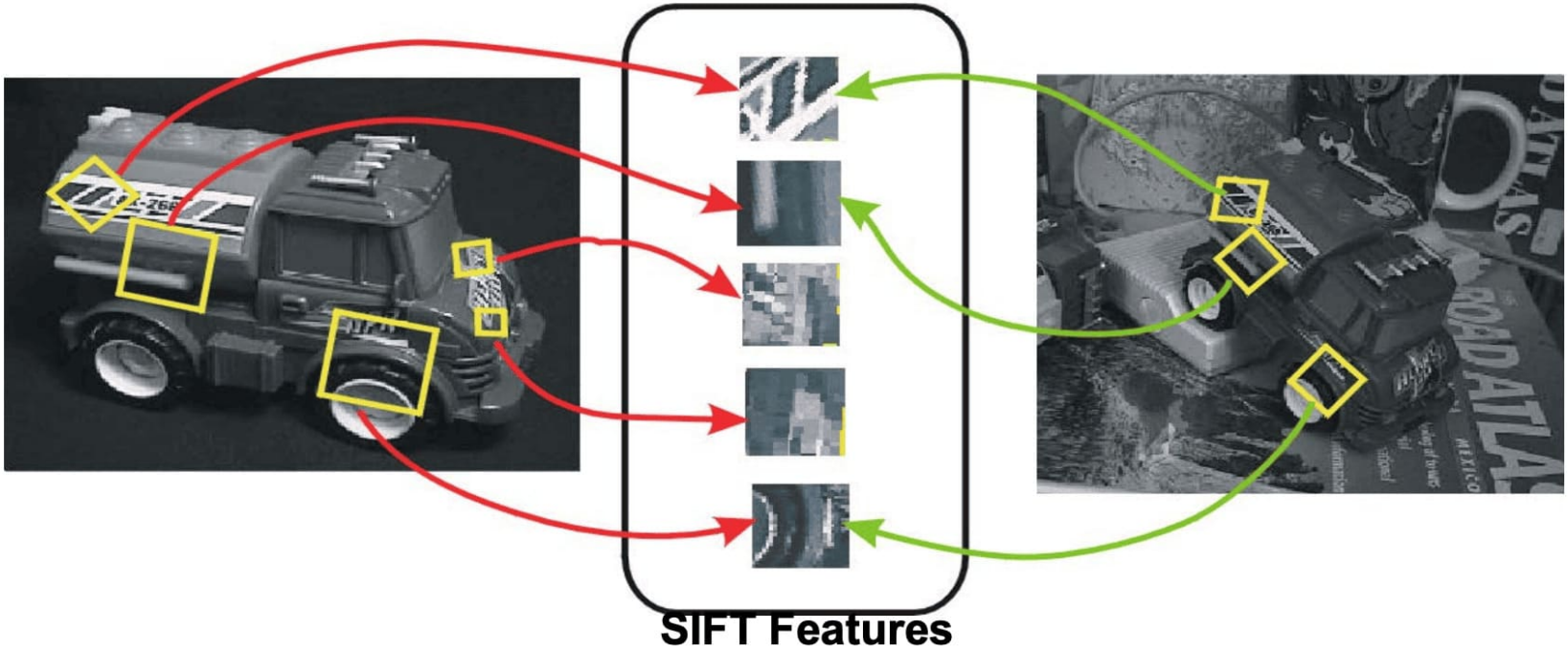
2 Pipeline
- Scale Space
- Detección de Extremos Locales
- Refinamiento de Precisión Subpíxel
- Eliminación de Respuestas en Bordes
- Asignación de Orientación
- Generación Del Descriptor
- Resumen del Pipeline

3 Aplicación
- Solución De Parámetros Afines

4 Conclusión

SIFT: Scale-Invariant Feature Transform

- Desarrollado por David Lowe en 1999, con versión definitiva en 2004.
- Algoritmo fuertemente heurístico.
- Detectores simples fallan ante cambios drásticos de iluminación, rotación, punto de vista o escala.
- SIFT transforma los datos de la imagen en coordenadas locales extrayendo *keypoints* que son altamente distintivos e invariantes a la escala y rotación.
- Algunos usos comunes: reconocimiento de objetos, reconstrucción 3D, emparejamiento de imágenes, etc.

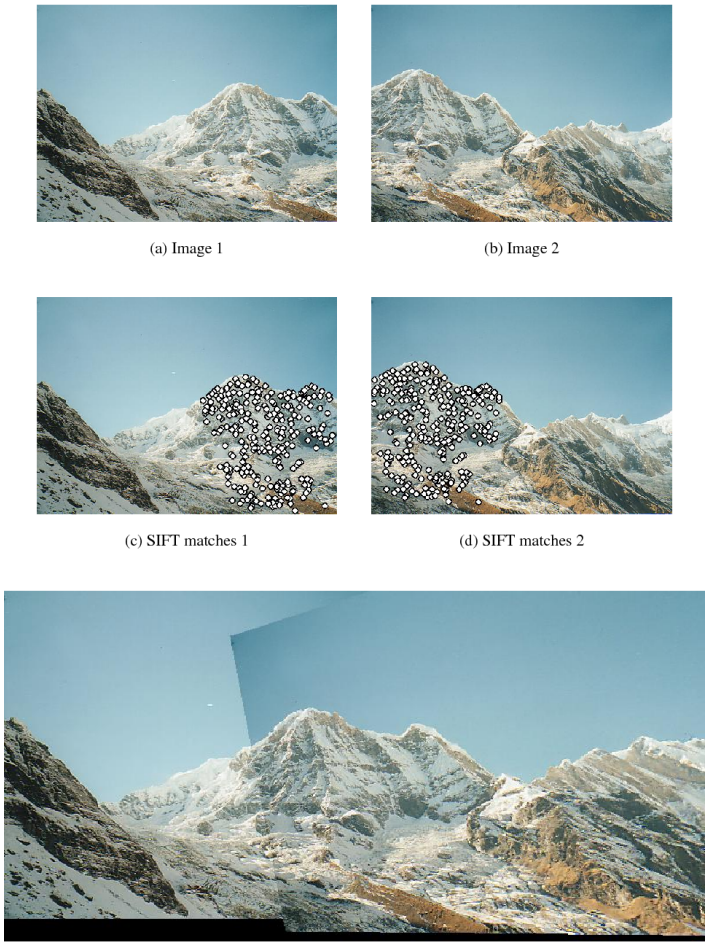


Introduction to SIFT (Scale Invariant Feature Transform). Medium (2019).

Figure 1: Extracción de características locales (*keypoints*).

SIFT Algorithm: How Feature Matching Works. Ultralytics (2025).

Figure 2: Emparejamiento de características (*Feature Matching*).



Recognising Panoramas. Brown y Lowe (2003).

Figure 3: Stitching de imágenes.

Algoritmo SIFT

1. Construcción del Espacio de Escala
2. Detección de Extremos Locales
3. Localización de keypoints
4. Filtrado de Puntos Inestables
5. Asignación de Orientación
6. Generación del Descriptor de Características

Scale Space

- Simula la pérdida de detalle al alejarse.
- *Filtro Gaussiano*: Lindberg demostró que es el único kernel lineal e invariante a traslaciones capaz de lograr esto sin introducir artefactos.

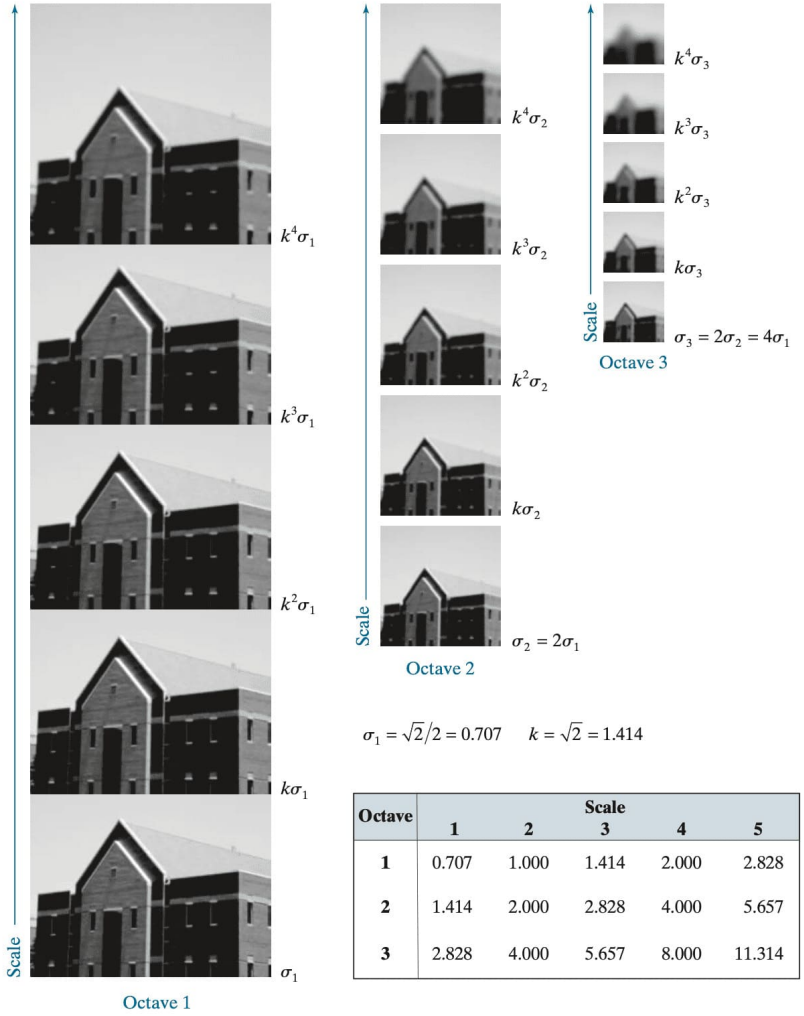
Espacio de escala de una imagen de grises:

$$L(x, y, \sigma) = G(x, y, \sigma) * f(x, y)$$

donde f es la imagen, $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ el filtro Gaussiano con σ el **parámetro de escala** que controla el suavizado de la imagen.

Scale Space

- **Octavas**: Bloques del espacio de escala donde el valor de desenfoque σ se duplica y el tamaño de la imagen se reduce a la mitad.
- **Intervalos** (s): Subdivisiones progresivas dentro de cada octava. Se generan obligatoriamente $s + 3$ imágenes suavizadas por octava por una necesidad matemática:
  - s intervalos para buscar extremos.
  - + 2 escalas de margen para comparar en 3D.
  - + 1 imagen extra necesaria para poder efectuar las restas.
- Para pasar a la siguiente octava ahorrando cálculos, se toma la tercera imagen desde el tope de la octava anterior, que ya posee exactamente el doble de σ, y se submuestrea por 2.



$\sigma_1 = \sqrt{2}/2 = 0.707 \quad k = \sqrt{2} = 1.414$

Octave	Scale 1	2	3	4	5
1	0.707	1.000	1.414	2.000	2.828
2	1.414	2.000	2.828	4.000	5.657
3	2.828	4.000	5.657	8.000	11.314

Digital Image Processing. Gonzalez y Woods (2018).

Detección de Extremos Locales

Queremos identificar puntos de la imagen que se destaquen de su entorno y que podamos volver a encontrar aunque la imagen cambie de escala o rotación. A esos puntos los llamamos **keypoints**.

En este paso vamos a obtener un primer conjunto de **keypoints candidatos**. Para eso:

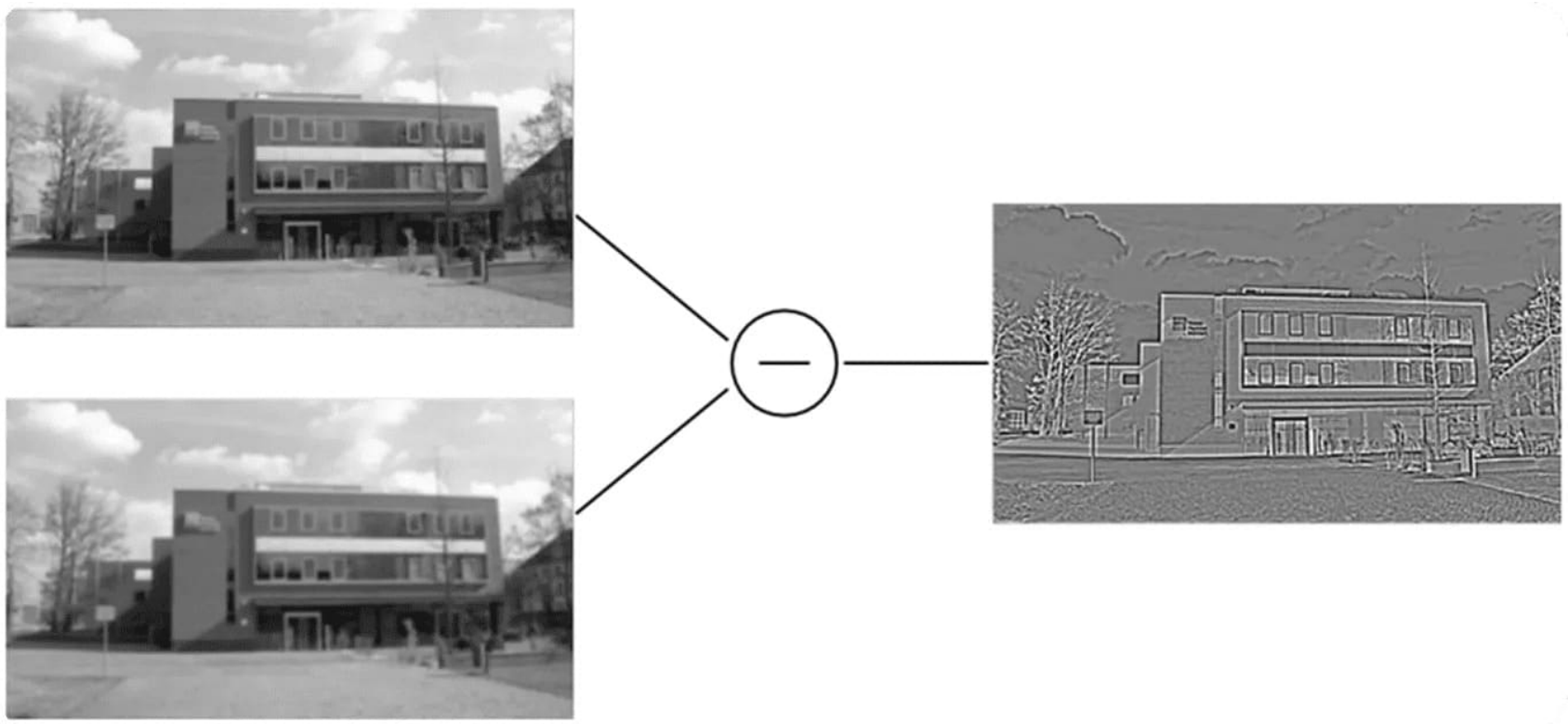
1. Calculamos la **Diferencia de Gaussianas** (DoG) a partir del espacio de escala.
2. Buscamos **extremos locales** en las DoGs.

Diferencia de Gaussianas (DoG)

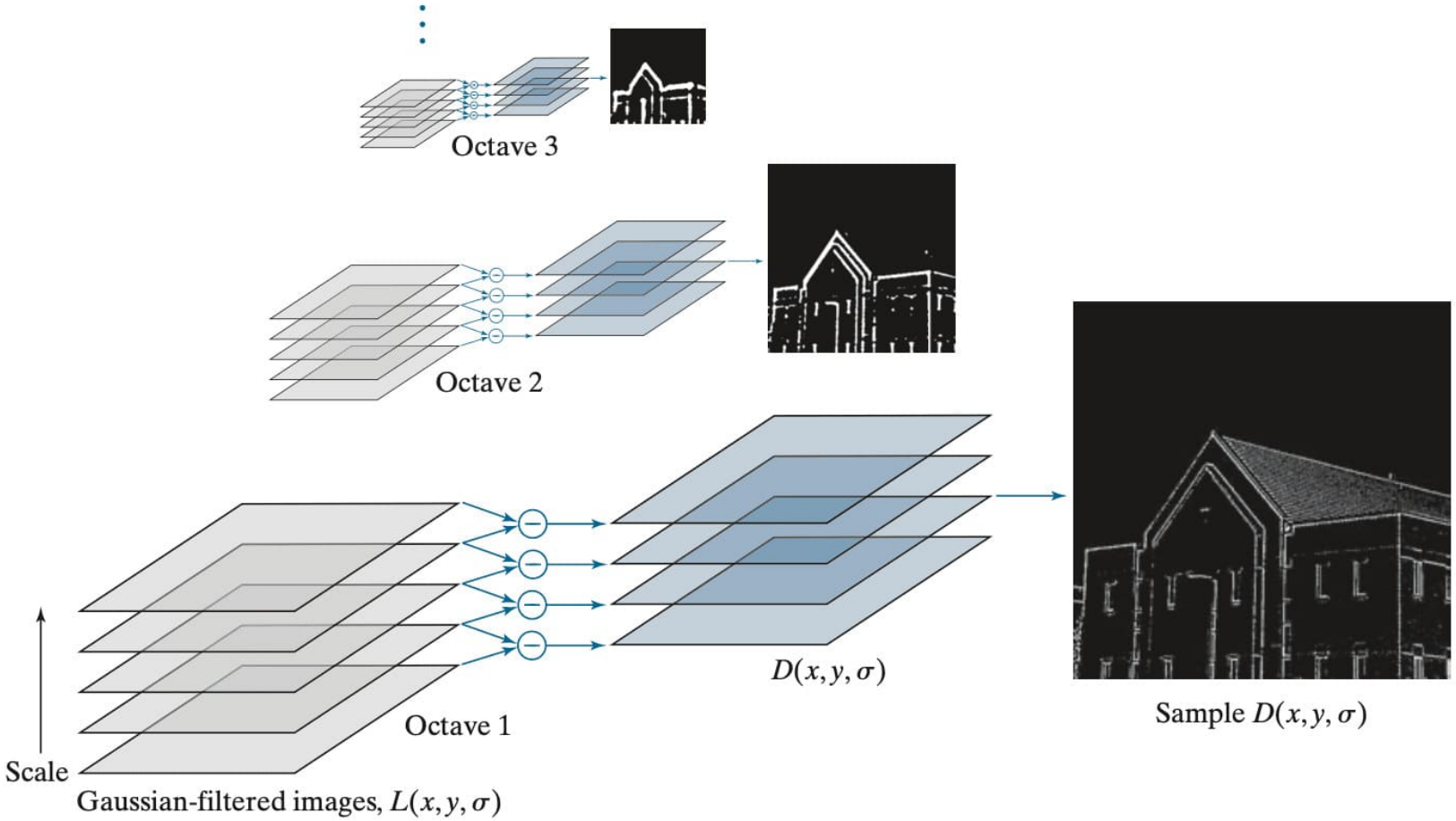
Se calcula restando imágenes suavizadas adyacentes dentro de una misma octava:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

- Se puede demostrar que $D \approx (k-1)\,\sigma^2 \nabla^2 G * f$, es decir, una aproximación al **Laplaciano de Gaussiana escala-normalizado**. Lindeberg mostró que los extremos de esa función son los puntos genuinamente **invariantes a la escala**.
- Computacionalmente es muy barata: solo requiere una resta, mientras que calcular el LoG explícitamente exigiría una segunda derivada.
- Funciona como un **detector de blobs**: en cada DoG, los píxeles con respuesta alta son aquellos donde la imagen tiene una estructura local del tamaño de σ.



Parallel Implementation Strategies for Hierarchical Non-uniform Memory Access Systems by Example of the Scale-Invariant Feature Transform Algorithm. Plauth et al. (2016).



Digital Image Processing. Gonzalez y Woods (2018).

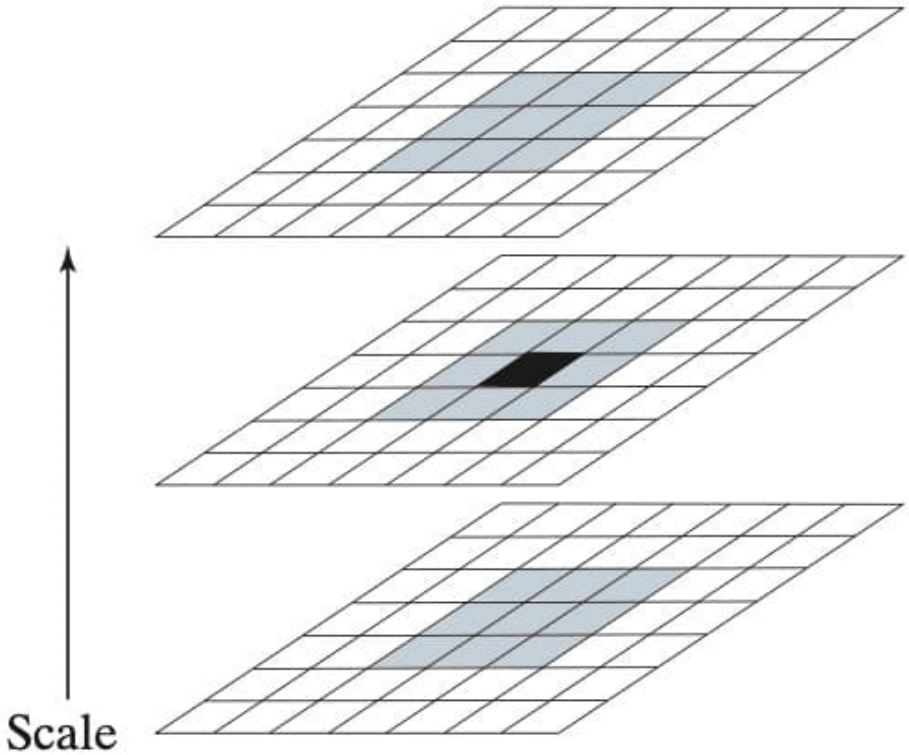
Detección de Extremos Locales en las DoGs

Buscamos los keypoints candidatos como **extremos locales en el espacio 3D** (x, y, σ). Cada píxel de una DoG se compara con sus **26 vecinos**:

- 8 vecinos en la misma escala,
- 9 en la escala superior,
- 9 en la escala inferior.

Se marca como **keypoint candidato** si es estrictamente mayor o menor que los 26.

Observación: las DoGs del borde de la octava (la primera y la última) no son buscables, porque les falta una escala vecina arriba o abajo. Por eso una octava con $s + 2$ DoGs solo permite buscar extremos en los s del medio.



Corresponding sections of three contiguous $D(x, y, \sigma)$ images

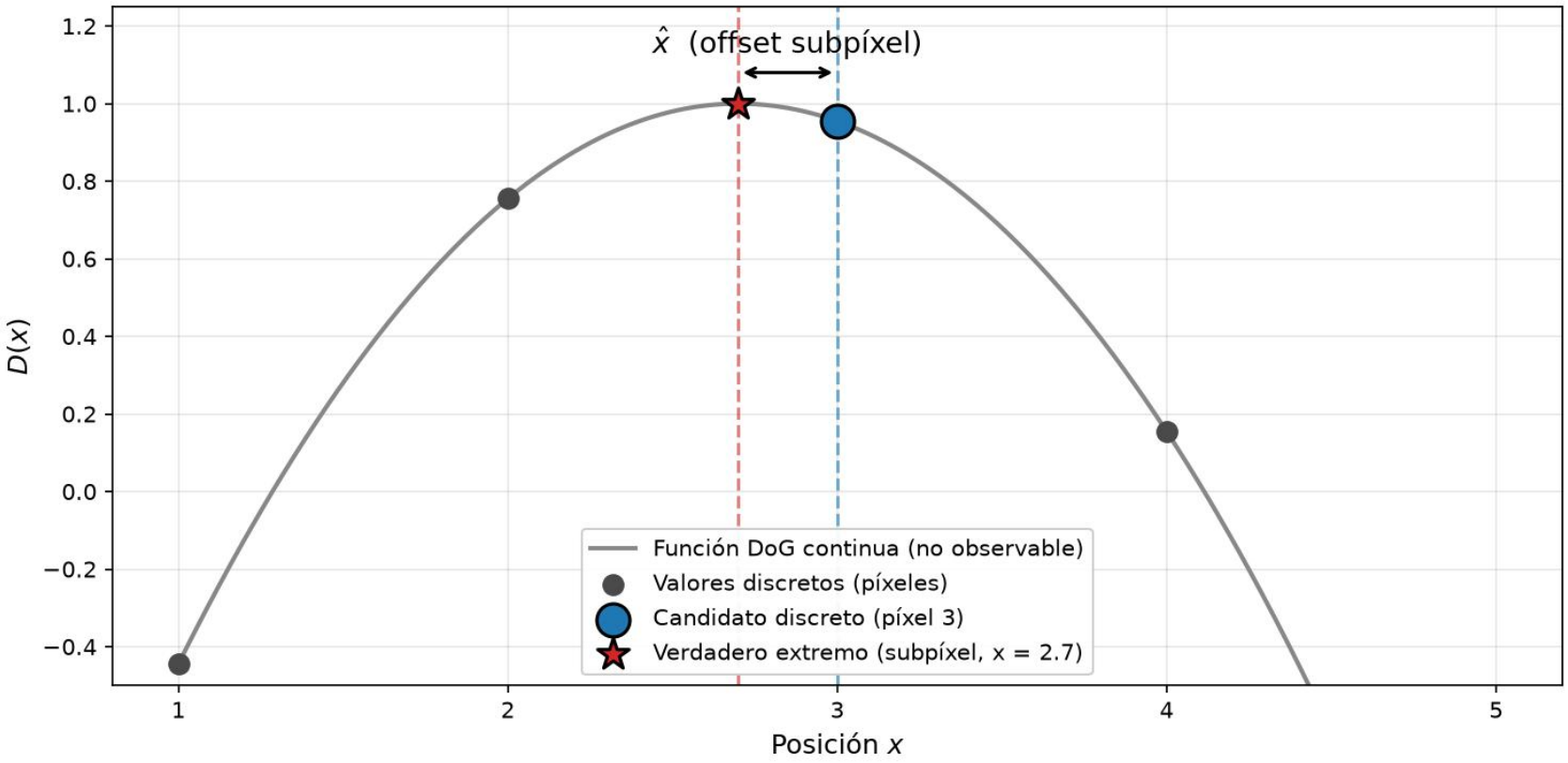
Digital Image Processing. Gonzalez y Woods (2018).

Refinamiento de Precisión Subpíxel

En el paso anterior detectamos los keypoints candidatos como extremos de la función DoG *en posiciones discretas* (los píxeles enteros). Pero la función DoG es continua: el **verdadero extremo** puede caer *entre* píxeles.

En este paso vamos a refinar la ubicación de cada candidato con **precisión subpíxel**. Para eso:

1. Aproximamos la función DoG localmente con una expansión de **Taylor de segundo orden** alrededor del candidato.
2. Calculamos el **offset** $\hat{x}$ que lleva del candidato al extremo real.
3. Si el offset es muy grande, iteramos desde el píxel vecino.
4. Descartamos el punto si la respuesta DoG en el extremo es muy débil.


$\hat{x}$ (offset subpíxel)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
$D(x)$
1
2
3
4
5
Posición x
Función DoG continua (no observable)
Valores discretos (píxeles)
Candidato discreto (píxel 3)
Verdadero extremo (subpíxel, x = 2.7)

Refinamiento de Precisión Subpíxel

Aproximamos la función DoG por su expansión de Taylor de segundo orden alrededor del candidato discreto x_0:

$$D(\mathrm{x}_0 + \mathrm{x}) \approx D(\mathrm{x}_0) + \nabla D(\mathrm{x}_0)^T \mathrm{x} + \frac{1}{2} \mathrm{x}^T \mathrm{H}(\mathrm{x}_0) \mathrm{x}$$

donde $\mathrm{x} = (x, y, \sigma)^T$ es el offset relativo al candidato, ∇D es el **gradiente** (vector de primeras derivadas) y H es la **matriz Hessiana 3D** (matriz de segundas derivadas), ambos evaluados en x_0:

$$\nabla D = \begin{pmatrix} D_x \\ D_y \\ D_\sigma \end{pmatrix}, \quad \mathrm{H} = \begin{pmatrix} D_{xx} & D_{xy} & D_{x\sigma} \\ D_{xy} & D_{yy} & D_{y\sigma} \\ D_{x\sigma} & D_{y\sigma} & D_{\sigma\sigma} \end{pmatrix}$$

donde $D_x = \partial D / \partial x$, $D_{xx} = \partial^2 D / \partial x^2$, $D_{xy} = \partial^2 D / \partial x \partial y$, etc.

Refinamiento de Precisión Subpíxel

Para hallar el extremo del polinomio, derivamos respecto a x e igualamos a cero:

$$\nabla D(\mathrm{x}_0) + \mathrm{H}(\mathrm{x}_0)\,\mathrm{x} = 0$$

$$\Longrightarrow \boxed{\hat{\mathrm{x}} = -\mathrm{H}(\mathrm{x}_0)^{-1}\,\nabla D(\mathrm{x}_0)}$$

El vector $\hat{\mathrm{x}}$ da el **offset subpíxel** del verdadero extremo respecto al candidato.

Refinamiento de Precisión Subpíxel

Una vez calculado $\hat{x}$, hay que verificar que el resultado tenga sentido:

- Si $|\hat{x}_i| \leq 0.5$ en todas las componentes, el extremo cae dentro del píxel candidato. La posición refinada del keypoint es

$$x_{\text{final}} = x_0 + \hat{x}$$

- Si **alguna componente supera** 0.5, significa que un píxel vecino está más cerca del verdadero extremo. En ese caso, el algoritmo **traslada el candidato a ese vecino y reitera** la interpolación desde ahí.

Además, se descartan los puntos con **contraste bajo**: evaluando el valor de la función en el extremo refinado,

$$D(x_0 + \hat{x}) = D(x_0) + \frac{1}{2} \nabla D(x_0)^T \hat{x},$$

si $|D(x_0 + \hat{x})| < 0.03$ el punto se descarta por ser propenso al ruido.

Eliminación de Respuestas en Bordes

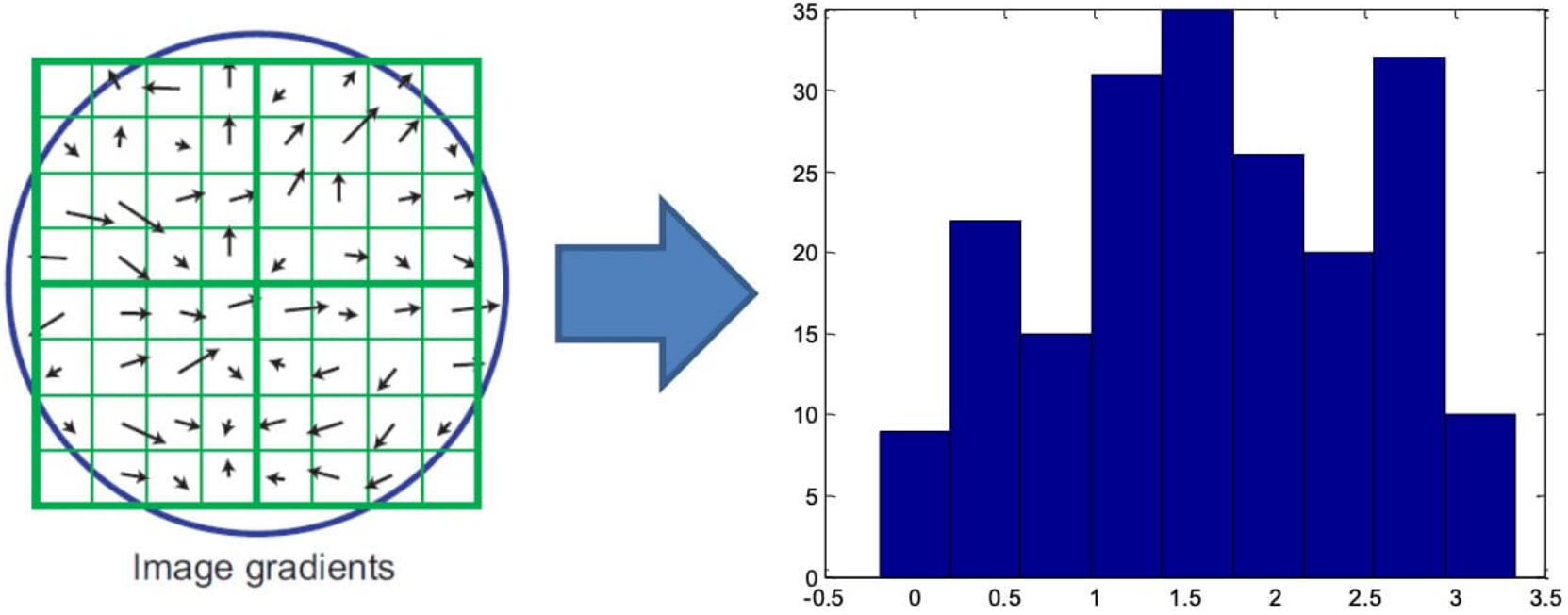
- Las DoG reaccionan a los bordes, pero estos son inestables. SIFT busca características tipo "esquina" bien localizadas
- Se evalúa la curvatura local mediante la matriz Hessiana 2×2 para las coordenadas espaciales (x, y), cuyos autovalores son directamente proporcionales a estas curvaturas, pues los bordestienen alta curvatura en un solo eje.
- Para evitar el lento cálculo de los autovalores, se comprueba la relación entre la Traza (Tr) y el Determinante (Det) de la matriz Hessiana H:

$$\frac{\mathrm{Tr}(\mathrm{H})^2}{\mathrm{Det}(\mathrm{H})} < \frac{(r+1)^2}{r}$$

- Lowe propone un límite experimental de $r = 10$. Si la relación supera este valor, el punto se descarta por comportarse como un borde.

Asignación de Orientación

- Para lograr una invarianza completa a las rotaciones de la imagen, se le asigna una orientación consistente a cada *keypoint* basándose en las propiedades locales de su entorno.
- Utilizando la imagen suavizada L más cercana a la escala del punto, se calculan la magnitud $M(x, y)$ y el ángulo $\theta(x, y)$ de todos los gradientes vecinos.
- Se construye un histograma de orientaciones dividido en 36 bins (cubriendo los 360°). Cada píxel vecino emite un voto que es ponderado por su magnitud de gradiente y por una ventana Gaussiana circular.
- El pico más alto del histograma define la orientación definitiva del punto. Si existe otro pico que alcance al menos el 80% del principal, se crea un segundo keypoint en la misma posición y escala con esta nueva orientación, lo que mejora significativamente la precisión del emparejamiento.



Digital Image Processing. Gonzalez y Woods (2018).

Generación Del Descriptor: Objetivo

Ya tenemos un punto estable con:

$$(x, y), \qquad \sigma, \qquad \theta$$

Objetivo Del Descriptor

Convertir el vecindario del keypoint en una "huella digital" numérica que sea:

- **distintiva**: dos lugares distintos deberían tener vectores distintos;
- **estable**: pequeños cambios de iluminación, escala, rotación o punto de vista no deberían destruirla;
- **comparable**: se pueda medir con distancia Euclidiana contra miles de descriptores.

En lugar de intensidades crudas, se usan **distribuciones locales de gradientes**.

Sistema De Coordenadas Local

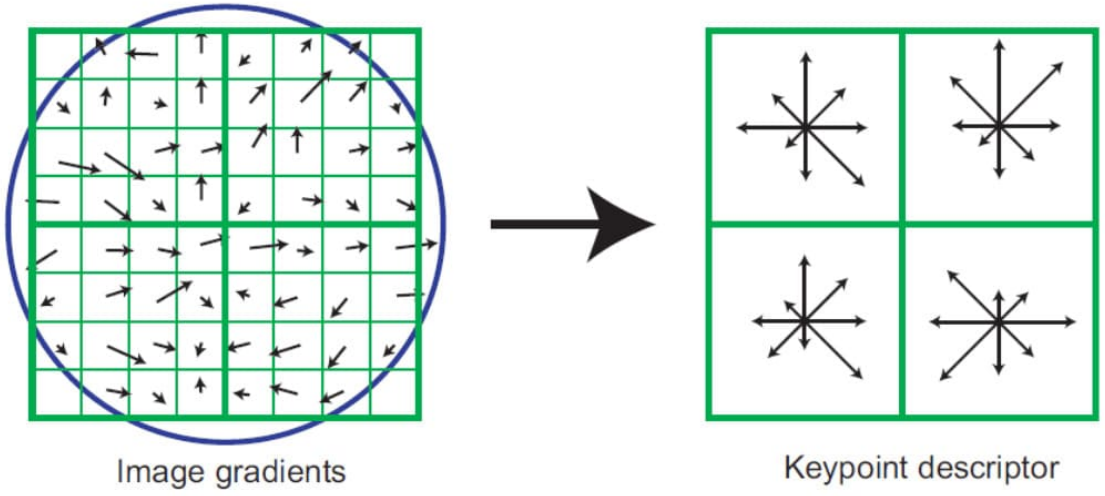
- El descriptor se calcula en la imagen suavizada correspondiente a la escala σ.
- Se toma una ventana alrededor del keypoint y se expresa todo en coordenadas **relativas** al punto.
- Luego se rota la ventana por $-\theta$:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

Si la imagen gira, el keypoint gira con ella, pero el descriptor se vuelve a expresar en el mismo marco local. Por eso los vectores siguen siendo comparables.

Es como describir una pieza mirando siempre "hacia adelante" según su propia orientación, no según la orientación de la foto.

De Un Parche A Histogramas



Descriptor local basado en histogramas de gradientes.

- Los gradientes del vecindario se agrupan en celdas espaciales.
- Cada celda acumula un histograma de orientaciones y todos los histogramas se concatenan.
- Las flechas no representan píxeles individuales: resumen cuánta "evidencia de borde" hay en cada dirección.

El Vector De 128 Dimensiones

En la versión clásica de Lowe:

$$\underbrace{4 \times 4}_{\text{subregiones espaciales}} \times \underbrace{8}_{\text{orientaciones}} = 128$$

- Se toma una región de muestreo proporcional a la escala del punto (comúnmente visualizada como 16×16 muestras).
- Cada celda 4×4 acumula un histograma de 8 direcciones: $0°, 45°, \ldots, 315°$.
- El resultado se concatena como un vector:

$$d = (h_{1,1,1}, h_{1,1,2}, \ldots, h_{4,4,8}) \in \mathbb{R}^{128}.$$

- Este vector es lo que después se compara con otros descriptores usando distancia Euclidiana.

Pesos Gaussianos E Interpolación Trilineal

- **Peso Gaussiano**: los gradientes cerca del centro del keypoint pesan más que los del borde de la ventana.
  - Evita que un pequeño corrimiento del punto cambie mucho el descriptor.
  - Reduce influencia de estructuras vecinas que quizá no pertenecen al mismo detalle.
- **Interpolación trilineal**: cada muestra puede repartir su voto entre:
  - celdas espaciales vecinas en x,
  - celdas espaciales vecinas en y,
  - bins de orientación vecinos.

El descriptor no cambia de golpe cuando un gradiente cae justo en la frontera entre dos celdas o dos orientaciones.

Normalización E Iluminación

1. Se normaliza el vector a norma 1:

$$d \leftarrow \frac{d}{\|d\|_2}$$

2. Se recortan valores grandes:

$$d_i \leftarrow \min(d_i, 0.2)$$

3. Se vuelve a normalizar.

- Si toda la imagen aumenta el contraste, los gradientes crecen proporcionalmente: la normalización lo cancela.
- Si aparece saturación o un brillo local muy fuerte, el recorte evita que un único borde domine todo el descriptor.

Invariancias Y Limitaciones Del Descriptor

Robusto Frente A

- traslación local,
- escala,
- rotación,
- cambios suaves de iluminación,
- pequeñas deformaciones o cambios de punto de vista.

Puede Fallar Si

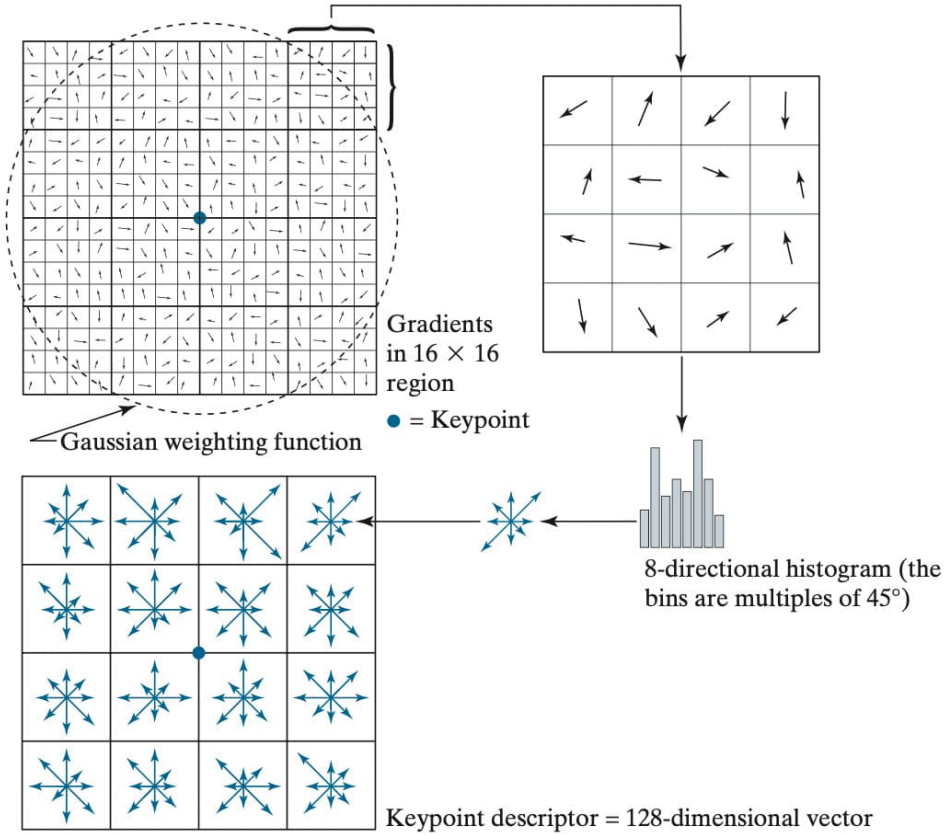
- hay muy poca textura,
- el objeto es repetitivo,
- cambia demasiado la perspectiva,
- hay desenfoque fuerte,
- el preprocesamiento destruye gradientes útiles.

Por eso, en aplicaciones reales se combina con **ratio test**, clustering por Hough y verificación geométrica.

Demo Del Descriptor En El Notebook

- En el notebook pueden ver cómo un keypoint pasa de ser un punto en la imagen a convertirse en un descriptor comparable:
  1. se recorta un parche alrededor del keypoint,
  2. se alinea con su orientación dominante,
  3. se divide en una grilla 4×4,
  4. y de cada celda sale un histograma de orientaciones.
- La idea no es memorizar la cuenta de 128 dimensiones, sino ver de dónde sale: 16 celdas, 8 orientaciones por celda.
- También pueden probar una versión con pasos que ya vimos antes: suavizado Gaussiano, umbralado, morfología, Canny y Hough antes de llegar a SIFT/matching.

Una imagen "más limpia" no siempre ayuda: lo importante es conservar gradientes estables, porque eso es lo que SIFT termina describiendo y comparando.



Digital Image Processing. Gonzalez y Woods (2018).

Figure 4: Cada keypoint queda representado por ubicación, escala, orientación y descriptor.

Resumen Del Pipeline SIFT

1. **Espacio de escala**: representación multiescala con filtros Gaussianos (octavas e intervalos).
2. **Extremos locales**: Diferencias de Gaussianas y comparación contra 26 vecinos en (x, y, σ).
3. **Refinamiento**: Taylor 3D para ubicar el extremo con precisión subpíxel y descartar bajo contraste.
4. **Filtrado**: eliminación de puntos tipo borde mediante la Hessiana espacial 2×2.
5. **Orientación**: histograma local de gradientes para fijar un marco de referencia rotacional.
6. **Descriptor**: vector de 128 dimensiones, normalizado, que resume la distribución local de gradientes.

Resumen: Embudo De Estabilidad

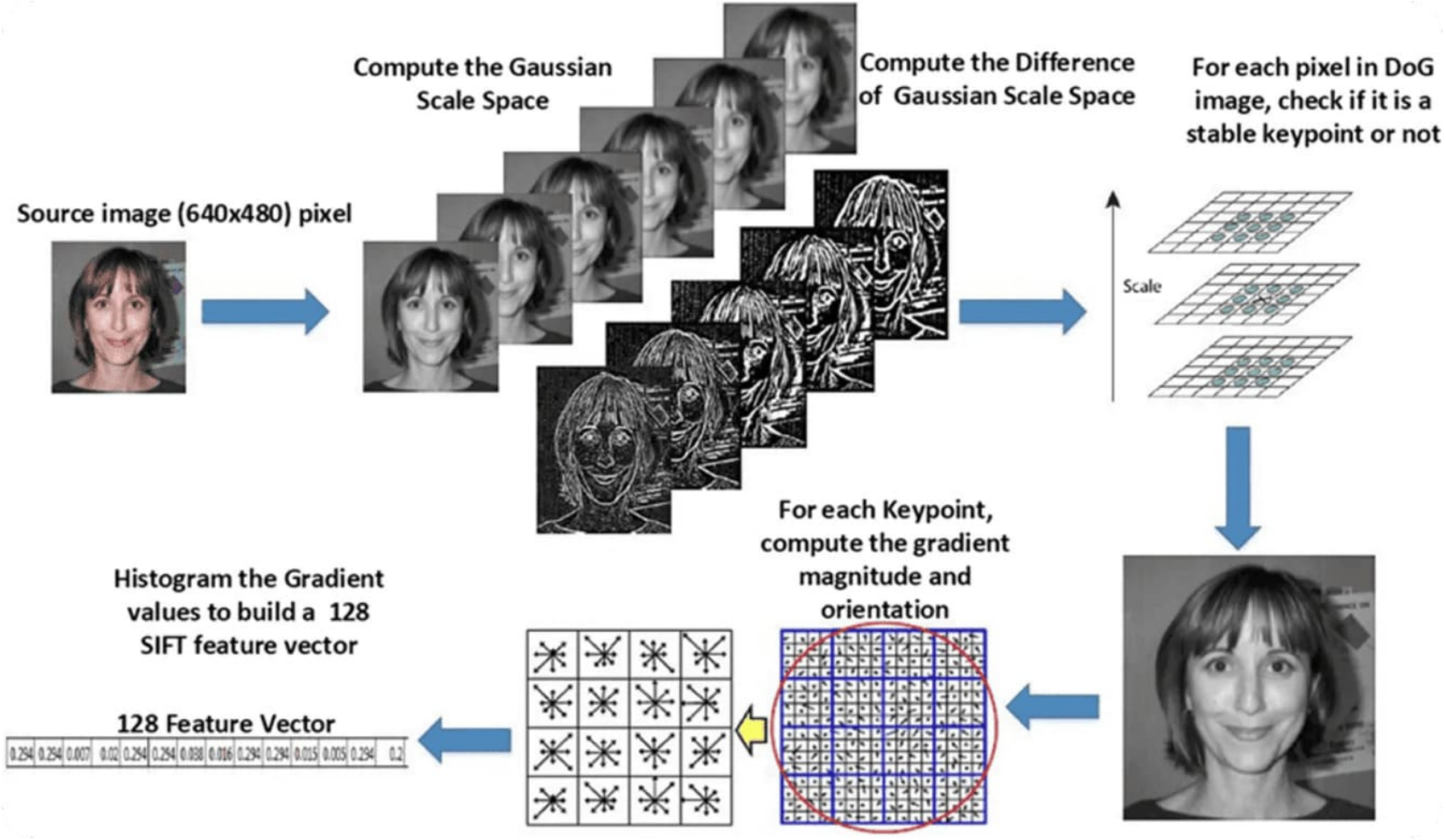
Etapa	Qué agrega	Qué descarta
DoG	candidatos multiescala	zonas sin estructura
Taylor	precisión subpíxel	bajo contraste
Hessiana	buena localización	bordes inestables
Orientación	invariancia a rotación	ambigüedad angular
Descriptor	comparación numérica	información irrelevante

SIFT no busca "puntos lindos": busca puntos **repetibles** y los convierte en vectores **comparables**.

Resumen: Conexión Con Temas Vistos

- **Filtros y convolución**: el espacio de escala se construye con filtros Gaussianos.
- **Canny / gradientes**: el descriptor también se apoya en magnitudes y orientaciones de gradiente, pero las acumula en histogramas locales.
- **Morfología y umbralado**: sirven como preprocesamiento o para limpiar máscaras/regiones de interés antes de detectar.
- **Transformada de Hough**: reaparece en reconocimiento de objetos para agrupar matches que votan por una misma pose.

El preprocesamiento debe mejorar la señal sin borrar las texturas y bordes locales que alimentan al descriptor.



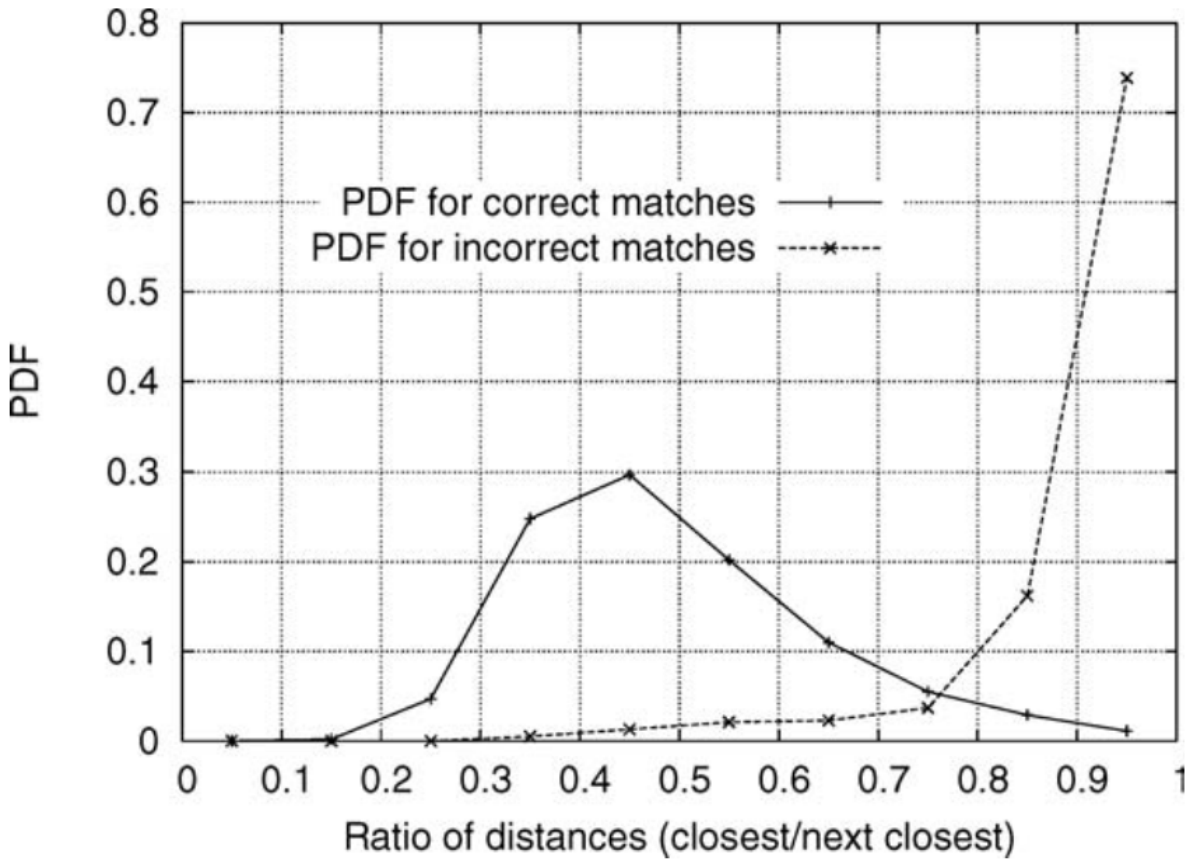
A novel SIFT architecture and ASIC implementation for real time SOC application. Qasaimeh et al. (2019).

Aplicación a Reconocimiento de Objetos

1. Reconocimiento de keypoints
2. Indexación eficiente de vecinos cercanos
3. Clustering con la Transformada de Hough
4. Solución de parámetros afines

Reconocimiento de objetos

- Búsqueda del Vecino Más Cercano: Para encontrar una coincidencia, se calcula la *Distancia Euclidiana* entre el vector del keypoint de la imagen original y los vectores de la nueva imagen. El vector con la distancia mínima es el candidato principal.
- El Problema del Ruido: En imágenes complejas, el fondo genera miles de emparejamientos falsos.
- **Ratio Test**: Para descartar los falsos positivos, se compara la distancia del 1° mejor candidato (d_1) contra la distancia del 2° mejor candidato (d_2).
  Si un punto es único, su mejor coincidencia estará significativamente más cerca que cualquier otra. Se rechazan todos los emparejamientos donde: $\frac{d_1}{d_2} > 0.8$.



Distinctive Image Features from Scale-Invariant Keypoints. Lowe (2004).

Figure 5: Función de Densidad de Probabilidad (PDF) de aciertos y errores evaluada sobre una base de datos de 40.000 puntos clave.

Búsqueda Eficiente en Alta Dimensionalidad

Para hacer matcheo, hay que comparar cada descriptor (vector de 128 elementos) contra miles de otros. **Hacerlo exhaustivamente es muy lento**.

Lo natural sería usar un **k-d tree**, pero con vectores tan largos **degenera** y termina siendo casi tan lento como comparar contra todos.

SIFT usa **Best-Bin-First (BBF)**: una variante del k-d tree que explora primero las regiones más prometedoras y corta después de revisar las primeras 200. Es *aproximada*: gana velocidad a cambio de equivocarse de vez en cuando.

Agrupación con Transformada de Hough

En imágenes muy desordenadas o con objetos tapados, los emparejamientos correctos pueden ser muy pocos.

- Agrupación (Clustering): Se usa la Transformada de Hough para que cada keypoint emparejado "vote" por las coordenadas donde cree que está el objeto.
- Contenedores de Votación: Los votos se acumulan en contenedores amplios para tolerar deformaciones no rígidas y errores de aproximación.
- Condición de Éxito: Si 3 o más características votan exactamente por la misma posición, se crea una hipótesis firme de que el objeto se encuentra allí.

Verificación Geométrica De Matches

Después del ratio test y de Hough todavía pueden quedar matches falsos.

Criterio Geométrico

Los matches correctos de un mismo objeto no deberían ser independientes: tienen que ser compatibles con una transformación global.

- Hough propone una hipótesis de pose: ubicación, escala y orientación aproximadas.
- Esta etapa la vuelve cuantitativa: ajusta la transformación que lleva puntos del modelo a puntos de la imagen.
- Si varios matches no obedecen esa transformación, se consideran **outliers**.

Un match aislado puede engañar; un conjunto que respeta la misma geometría es mucho más difícil de falsificar.

Modelo Afín

Una transformación afín 2D permite traslación, rotación, escala, estiramiento y cizalla:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- (x, y): coordenadas del keypoint en la imagen/modelo de entrenamiento.
- (u, v): coordenadas del keypoint matcheado en la imagen nueva.
- Incógnitas: $m_1, m_2, m_3, m_4, t_x, t_y$.

Por Qué Mínimo 3 Matches

Cada match aporta dos ecuaciones. Como hay 6 incógnitas, hacen falta al menos 3 matches no degenerados.

Sistema Lineal Por Mínimos Cuadrados

Para un match:

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

Con varios matches se apilan filas:

$$A\,p = b, \qquad \hat{p} = (A^T A)^{-1} A^T b$$

- Si hay exactamente 3 matches, el sistema queda justo determinado.
- Si hay más de 3, mínimos cuadrados busca la transformación que minimiza el error total.
- En código real se usa `np.linalg.lstsq` o funciones robustas como `estimateAffine2D`/`findHomography`.

Errores Residuales Y Outliers

Una vez estimada la transformación, se proyecta cada punto del modelo:

$$\hat{q}_i = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} p_i + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

y se mide el residuo:

$$e_i = \|q_i - \hat{q}_i\|_2.$$

- Si e_i es chico, el match es compatible con la hipótesis.
- Si e_i es grande, probablemente era un falso positivo.
- Lowe itera: elimina outliers, recalcula mínimos cuadrados y vuelve a verificar.

Aceptación Final Del Objeto

1. Se parte de un cluster con al menos 3 votos consistentes.
2. Se ajusta la transformación afín por mínimos cuadrados.
3. Se eliminan outliers según el error residual.
4. Si quedan menos de 3 inliers, se rechaza la hipótesis.
5. Si quedan suficientes, se evalúa la probabilidad de que sea una coincidencia falsa.

La interpretación se acepta si la probabilidad final de ser correcta supera el 98%.

Demo Afín En El Notebook Y Recursos

- El notebook incluye una celda que arma explícitamente la matriz A, resuelve $Ap = b$ y compara la transformación estimada contra la real.
- También se agrega una demo de matching con ratio test y estimación geométrica para visualizar inliers/outliers.

Recursos Online Para Probar

- IPOL Anatomy of SIFT: demo online con parámetros de SIFT.
- OpenCV Feature Matching + Homography: SIFT, ratio test y RANSAC.
- scikit-image SIFT: ejemplo reproducible con transformaciones afines.
- AutoStitch: panoramas construidos mediante matches invariantes.

Conclusión

- SIFT es sumamente eficaz para el emparejamiento de imágenes y reconocimiento de objetos en bases de datos.
- Funciona con precisión quirúrgica incluso si la subimagen sufrió rotaciones pronunciadas o reducciones drásticas de tamaño a la mitad.
- Aunque SIFT tolera ciertos cambios lumínicos, su tasa de éxito se optimiza radicalmente si se realiza una ecualización de histograma o mejora de contraste previa al procesamiento de la imagen.