

# **Recuperación de Información Multimedia**

## **Descriptores Locales**

**CC5213 – Recuperación de Información Multimedia**

Departamento de Ciencias de la Computación

Universidad de Chile

Juan Manuel Barrios – <https://juan.cl/mir/> – 2020

# Motivación

- ¿Cómo buscar la aparición de un objeto conocido en una escena de consulta?
  - Cómo calcular similitud parcial entre imágenes
- La búsqueda usando descriptores globales no permite encontrar imágenes que comparten una zona

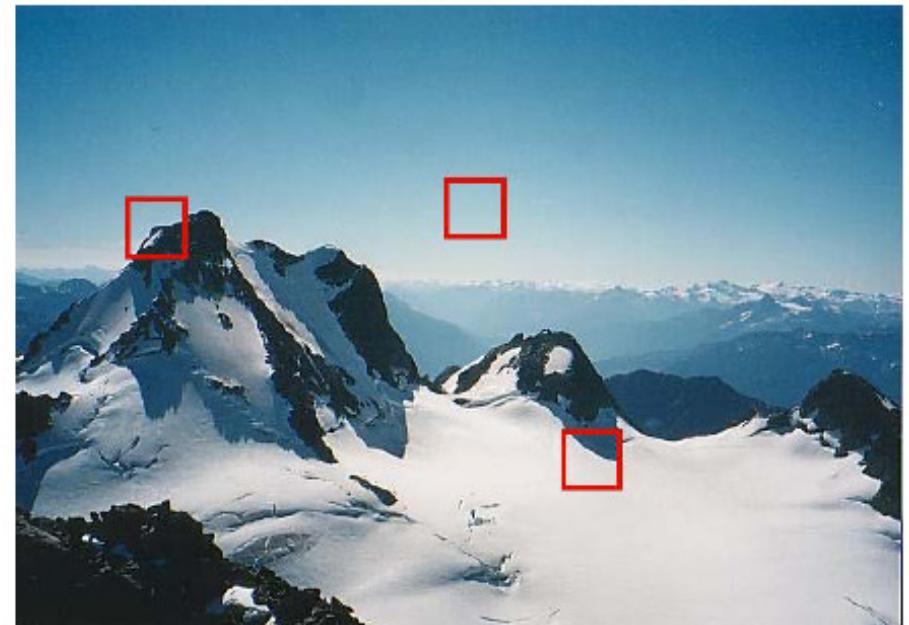
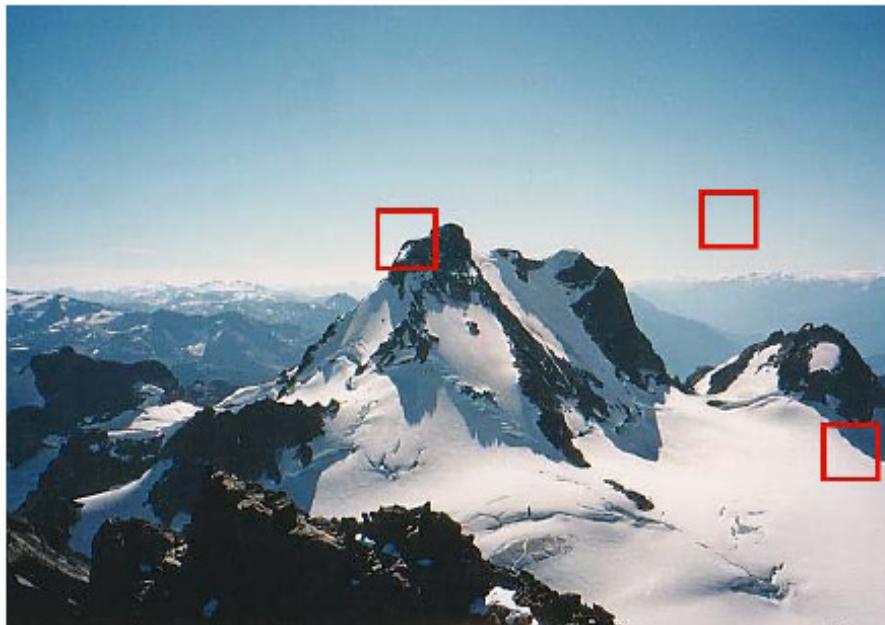




# Descriptores Locales

- Representar la imagen por varios descriptores, cada uno con información del contenido de una pequeña zona (patch) de la imagen
  - Dicho de otra forma: dividir la imagen en pequeños trozos independientes y calcular un descriptor para cada uno
- Las zonas a representar deben ser fácilmente identificables ante cambios de tamaño y perspectiva de los objetos en la foto
- Se espera que el descriptor de cada zona sea robusto a cambios de tamaño y perspectiva del contenido

# Puntos de Interés

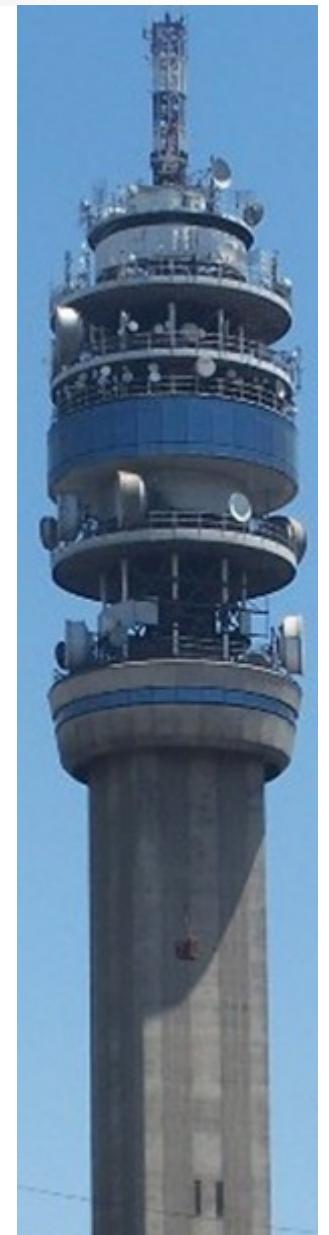


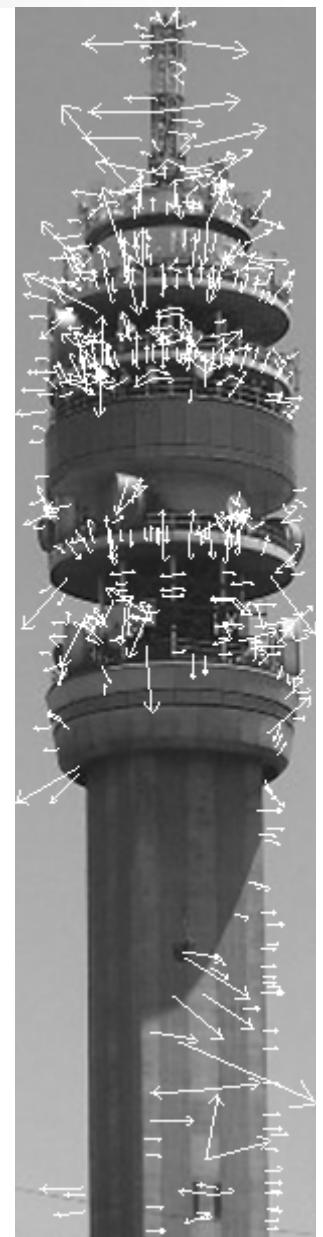
Ver Szeliski, cap 4.



# Puntos de Interés

- Punto de interés (keypoint) de una imagen:
  - Pixel que se puede distinguir de los pixeles que lo rodean
  - Usualmente los detectores son del tipo esquinas o manchas (blob)
- Detección invariante a:
  - Color, Rotación, Traslación, Escala
  - Otras transformaciones geométricas
    - Afines: rotaciones y deformaciones romboidales (manteniendo el paralelismo)
    - Homográficas: proyecciones



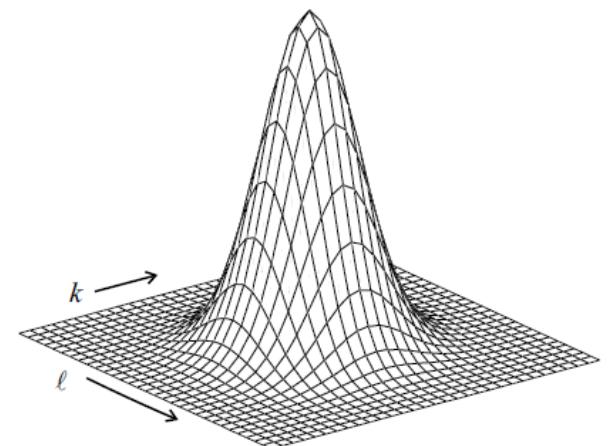


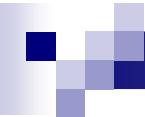
Ejemplo: SIFT (D.Lowe)

# Scale Space

- Para la invarianza a escala se utiliza el concepto de “Scale Space”
- Una imagen produce un scale space al hacer convolución con un kernel gaussiano de varianza  $\sigma$

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





# Scale Space

- La dimensión  $\sigma$  es continua entre 0 e infinito:

$$L(x, y, \sigma) = (I * G_\sigma)(x, y)$$

$$L(x, y, 0) = I(x, y)$$



$\sigma=5$



$\sigma=2$



$\sigma=1.4$



$\sigma=1$



$\sigma=0.7$

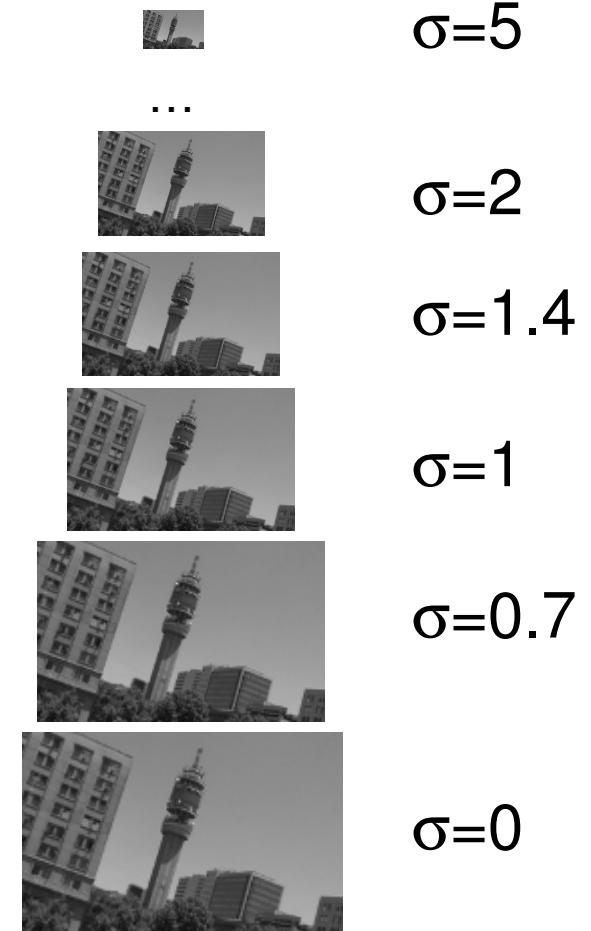


$\sigma=0$

# Pirámide de Gauss

- Como las imágenes con mayor  $\sigma$  contienen menos detalle, la imagen podría ser reducida sin perder información

- “Pirámide de Resolución”,  
“Pirámide de Gauss”,  
“Gaussian Pyramid”



# Multi-resolución

- Un pixel en una imagen pequeña corresponde a una zona en una imagen más grande
- Dos imágenes iguales pero de distinto tamaño, a partir de algún punto sus pirámides van a coincidir



$\sigma=5$





# **Detector de esquinas**

# Detector de Moravec (1981)

- Dada una imagen  $I$ , tomar una ventana pequeña de tamaño  $M \times M$  en torno a  $(x_0, y_0)$

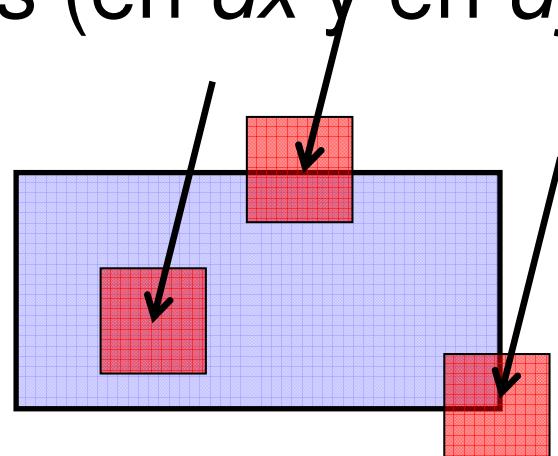
$$w(a, b) = \begin{cases} 1 & \text{si } a \in [-\frac{M}{2}, \frac{M}{2}] \wedge b \in [-\frac{M}{2}, \frac{M}{2}] \\ 0 & \text{si no} \end{cases}$$

- Elegir un desplazamiento  $(dx, dy)$
- Calcular la “autodiferencia” o “autocorrelación” entre las vecindades de  $(x_0, y_0)$  y  $(x_0+dx, y_0+dy)$

$$E(x_0, y_0, dx, dy) = \sum_{x,y} (I(x + dx, y + dy) - I(x, y))^2 \cdot w(x - x_0, y - y_0)$$

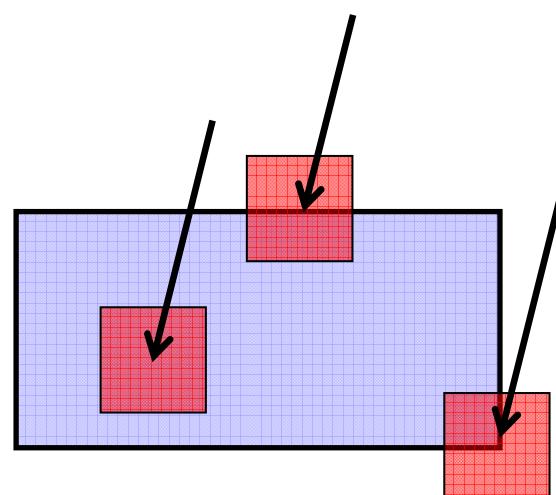
# Detector de Moravec

- En zonas planas no hay cambios
- En los bordes hay cambios sólo en una dirección (en  $dx$  o en  $dy$ )
- En las esquinas hay cambios en ambas direcciones (en  $dx$  y en  $dy$ )



# Detector de Moravec

- Para un  $(x_0, y_0)$  se calcula E para distintos  $(dx, dy)$  y se toma el mínimo
- Se eligen los puntos  $(x_0, y_0)$  que sean máximos locales de E



# Detector de Harris & Stephens (1988)

- Mejora al detector de Moravec
- La ventana  $w$  es una gaussiana
- En vez de usar  $I(x+dx, y+dy)$  se usa la aproximación de Taylor de primer orden:

$$I(x + dx, y + dy) = I(x, y) + \frac{\partial I}{\partial x}(x, y) \cdot dx + \frac{\partial I}{\partial y}(x, y) \cdot dy$$

$$E(x_0, y_0, dx, dy) = \sum_{x,y} \left( \frac{\partial I}{\partial x}(x, y) \cdot dx + \frac{\partial I}{\partial y}(x, y) \cdot dy \right)^2 w(x, y)$$

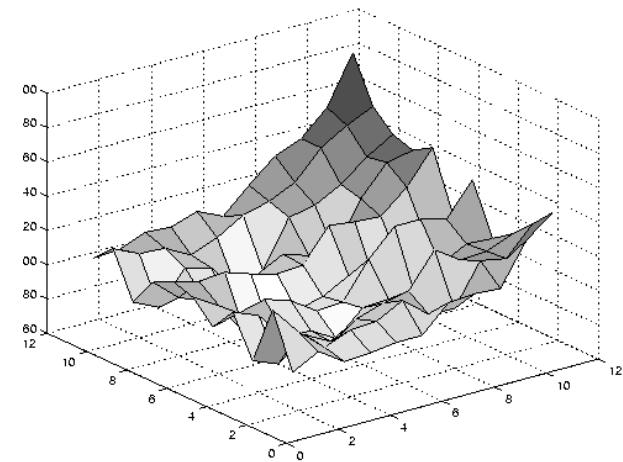
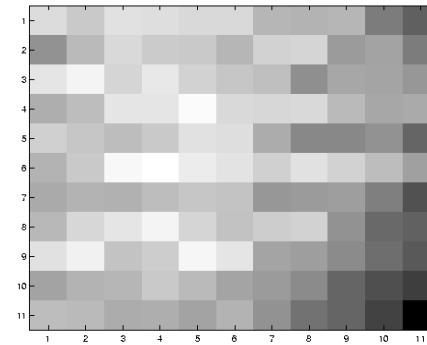
# Detector de Harris & Stephens

- El que puede ser escrito en forma de matriz, usando las derivadas parciales, llamado la matriz de auto-correlación:

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Se desea localizar pixeles que tengan un gradiente alto → sus dos valores propios  $\lambda_1$  y  $\lambda_2$  deben ser altos

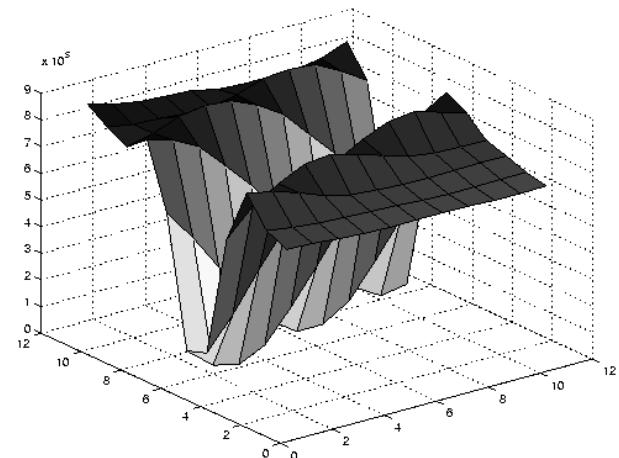
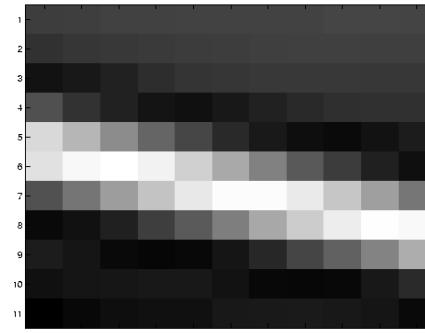
# Puntos de interés y valores propios



$\lambda_1$  y  $\lambda_2$  pequeños

Ver Szeliski, cap 4.

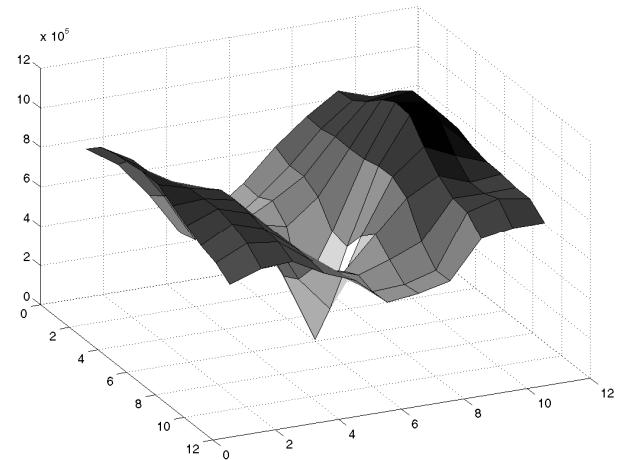
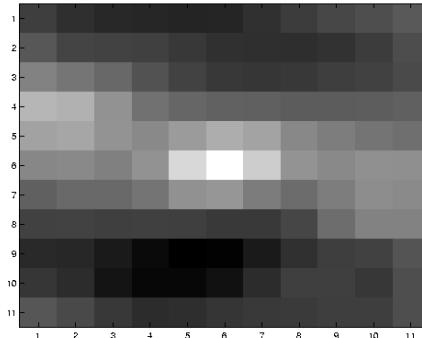
# Puntos de interés y valores propios



$\lambda_1$  alto,  $\lambda_2$  pequeño

Ver Szeliski, cap 4.

# Puntos de interés y valores propios



$\lambda_1$  y  $\lambda_2$  altos

Ver Szeliski, cap 4.

# Detector de Harris & Stephens

- Calcular los valores propios puede ser costoso
- Harris propone fijarse en los valores de  $\lambda_1 * \lambda_2$  y  $\lambda_1 + \lambda_2$  para maximizar:  
$$\text{determinante}(M) - \kappa^* \text{traza}(M)^2$$
  - $\kappa$  un parámetro ajustable.
  - Medida de “cornerness” de un punto
- Luego de un trabajo matemático...

# Detector de Harris & Stephens

- Resultado: marcar un punto de interés cuando la función  $g$  supera cierto umbral:

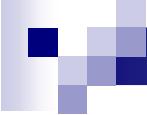
$$g = ab - c^2 - \alpha(a + b)^2$$

$$\alpha \in [0.04, 0.15]$$

$$a = \sum_{x,y} w(x,y) \frac{\partial I}{\partial x}(x,y)^2$$

$$b = \sum_{x,y} w(x,y) \frac{\partial I}{\partial y}(x,y)^2$$

$$c = \sum_{x,y} w(x,y) \frac{\partial I}{\partial x}(x,y) \frac{\partial I}{\partial y}(x,y)$$

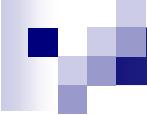


# Detector de Shi-Tomasi (1994)

- Basado en el de Harris
- En vez de maximizar la función  $g$ , lo que hay que maximizar es:  $\min(\lambda_1, \lambda_2)$
- Muestran experimentalmente que mejora a Harris



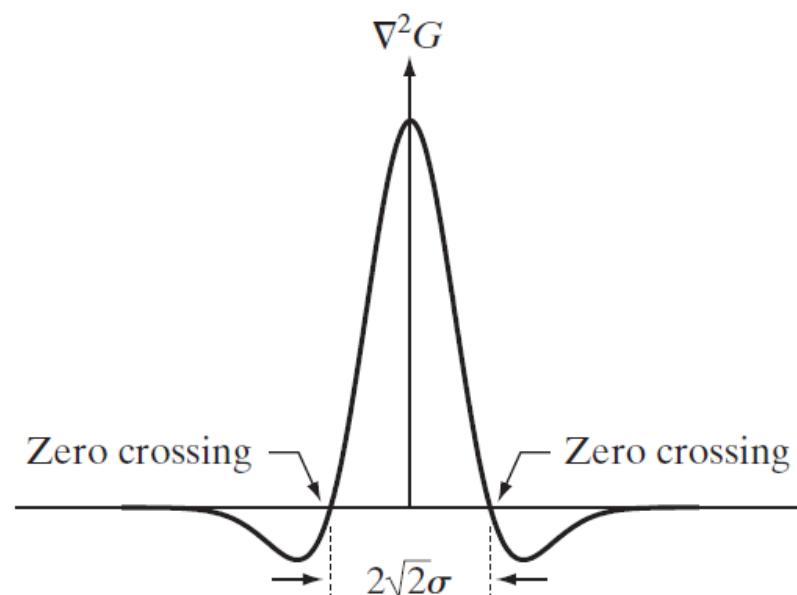
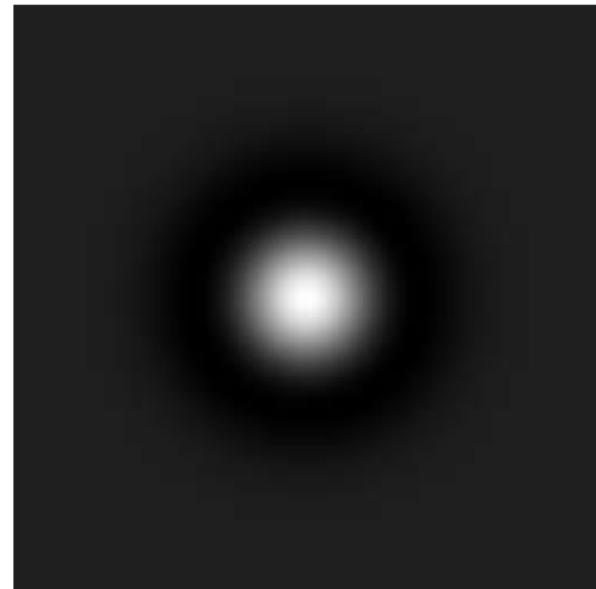
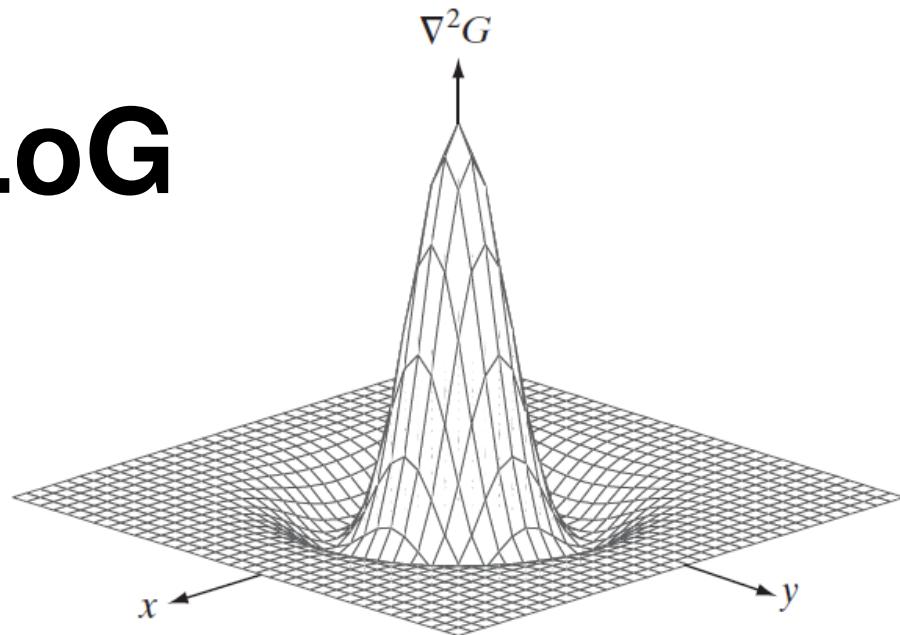
# **Detector de “Blobs”**



# Recuerdo: LoG

- El Laplaciano de Gaussiana (LoG) consiste en primero aplicar un suavizado gaussiano y luego el kernel Laplaciano
  - Útil para resaltar detalles y detectar bordes
- Puede ser aproximado por una Diferencia de Gaussianas (DoG)

# LoG



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

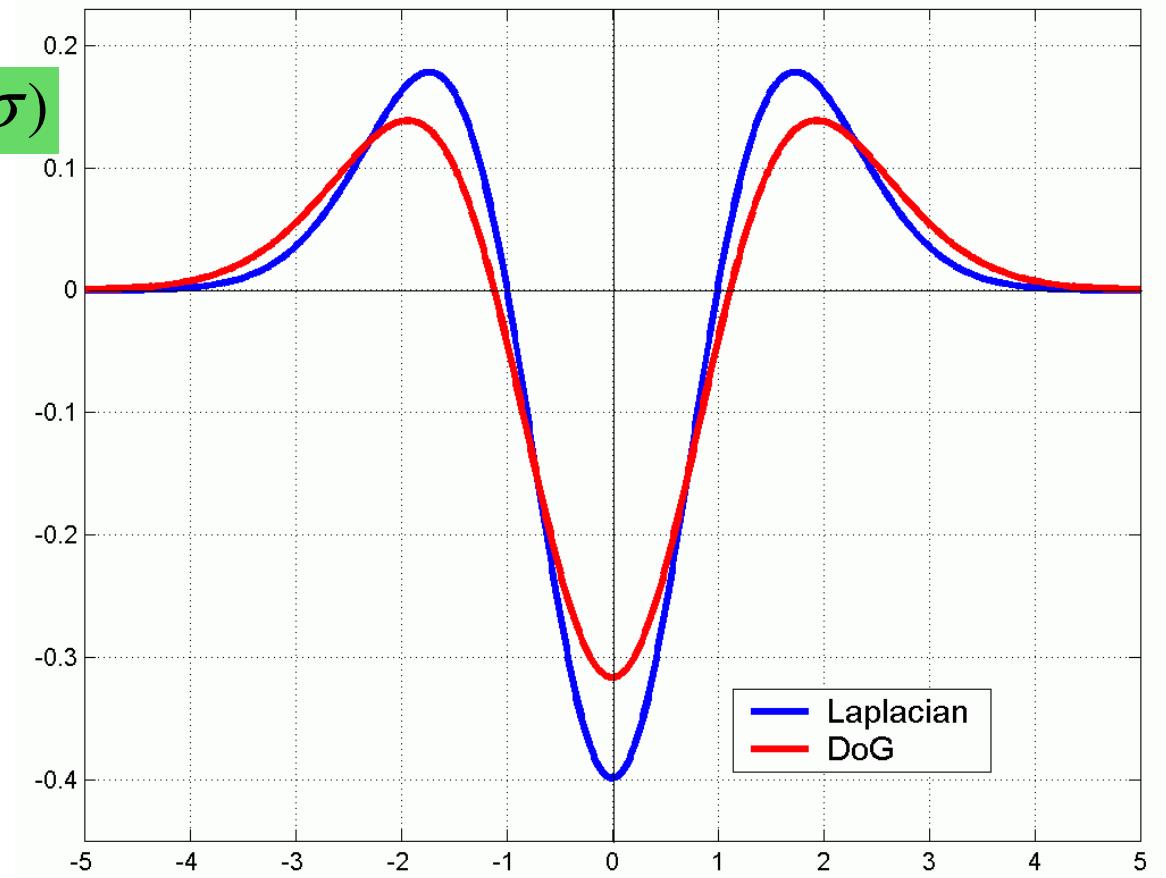
Ver Gonzalez, cap. 10

# Comparación DoG vs LoG

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$$

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

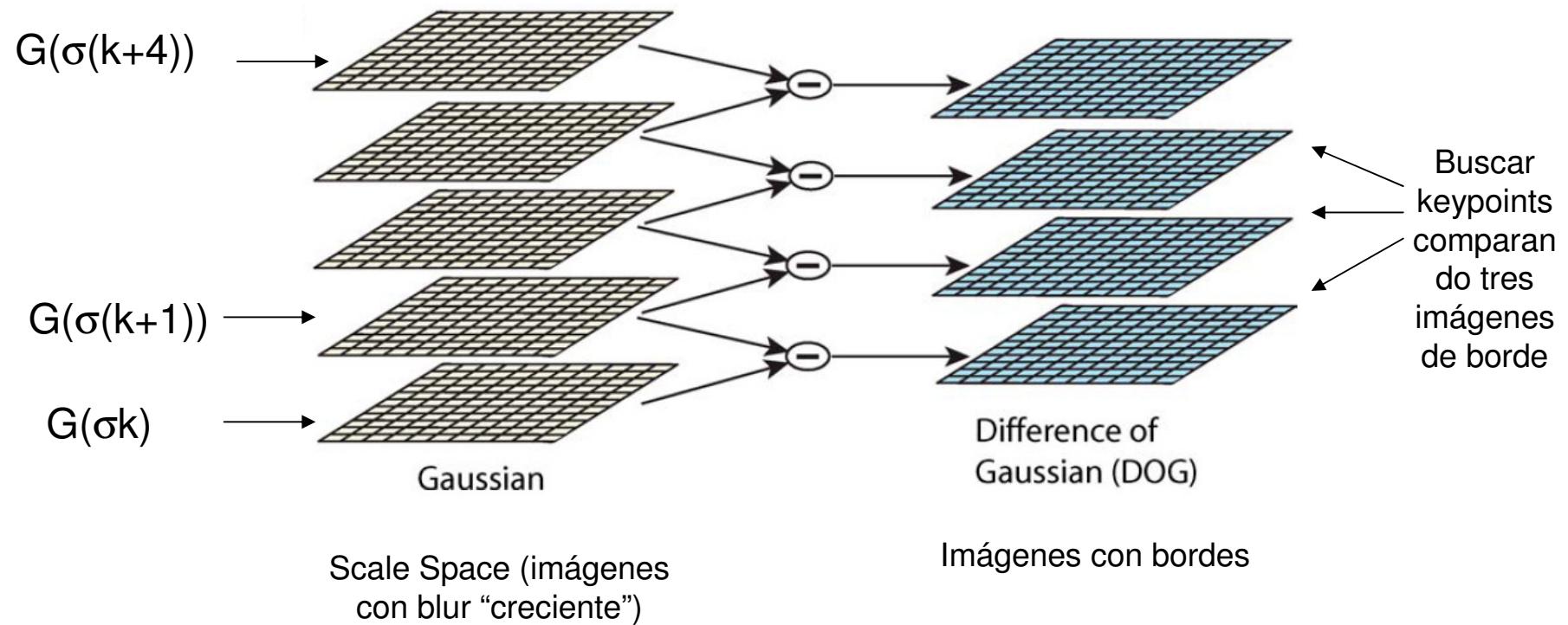
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



# Detección de Keypoints con DoG

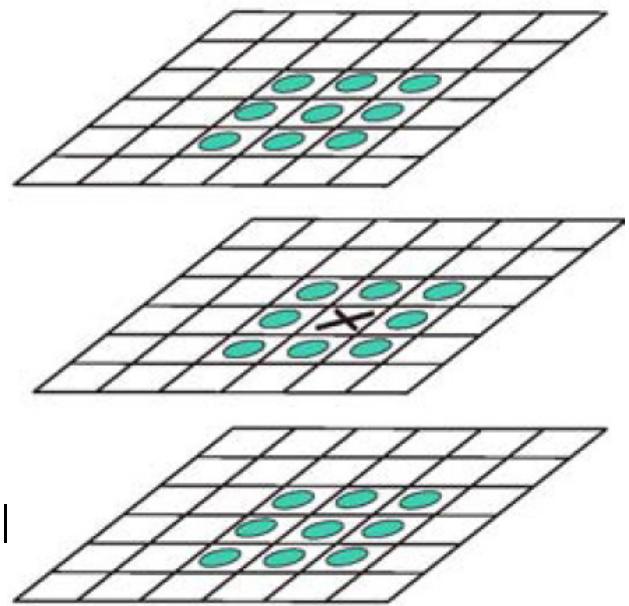
- Detecta pixeles “blobs” o manchas
  - Son zonas distintivas de la imagen
- Además de blobs tiende a detectar bordes, pero se eliminan después
- Usar el scale space para detectar keypoints de diferentes tamaños
  - 1<sup>a</sup> octava=imagen original, 2<sup>a</sup> octava=imagen reducida a la mitad, 3<sup>a</sup> octava=imagen reducida a un cuarto, etc.
- Laplacian Pyramid: Secuencia similar a la pirámide de gauss pero con secuencias a escala de bordes según LoG-DoG

# Diferencia de Gaussianas

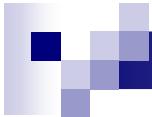


# Keypoints en DoG

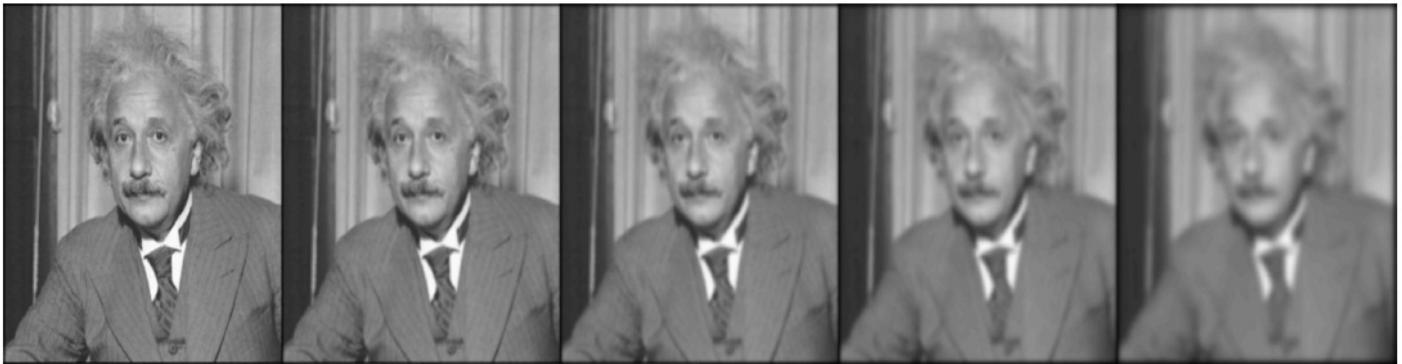
- Comparar cada pixel con sus 26 “vecinos” (8 vecinos en la misma imagen de borde, 9 píxeles de la imagen de borde anterior y 9 de la siguiente imagen de borde)
- Seleccionar píxeles mayores o menores que sus vecinos
  - Son píxeles de borde que desaparecen al aplicar una gaussiana (es decir, blobs o detalles)
- Se descartan candidatos con bajo contraste (gradiente bajo) o de borde (gradiente menor en un sentido)



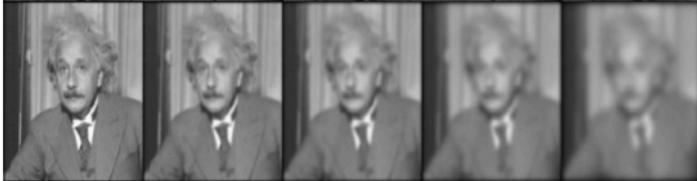
El pixel X es candidato a keypoint si es mayor o menor que sus 26 vecinos



1<sup>a</sup> octava →



2<sup>a</sup> octava →

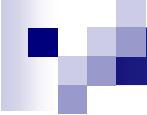


3<sup>a</sup> octava →



4<sup>a</sup> octava →





# Invariancia a rotaciones

- Hasta ahora, un punto de interés queda definido según su ubicación espacial ( $x, y$ ) y escala (tamaño)  $\sigma$
- Para ser invariante a rotaciones, se calculará una orientación fija  $\theta$  para cada keypoint según el contenido de la imagen:
  - Opción 1: calcular un eje normalizado de forma similar a PCA

# Invariancia a rotaciones

## ■ Opción 2 (propuesto por D.Lowe):

- Para todos los pixeles alrededor del keypoint se calcula la magnitud del gradiente ( $m$ ) y su ángulo ( $\theta$ ) ( $L$  es la imagen más la gaussiana  $\sigma$  correspondiente)

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y + 1) - L(x, y - 1)) / (L(x + 1, y) - L(x - 1, y)))$$

- Se selecciona el  $\theta$  mayoritario para la zona
- Si hay dos ángulos mayoritarios se duplica el keypoint, cada uno con un ángulo distinto

# SIFT (D.Lowe)

- Un keypoint se representa por  $(x, y, \sigma, \theta)$
- Para representar la vecindad de cada keypoint (patch) se calcula un histograma de orientaciones de gradiente de 4x4 zonas y 8 bins
  - Orientar el patch según el  $\theta$  del keypoint

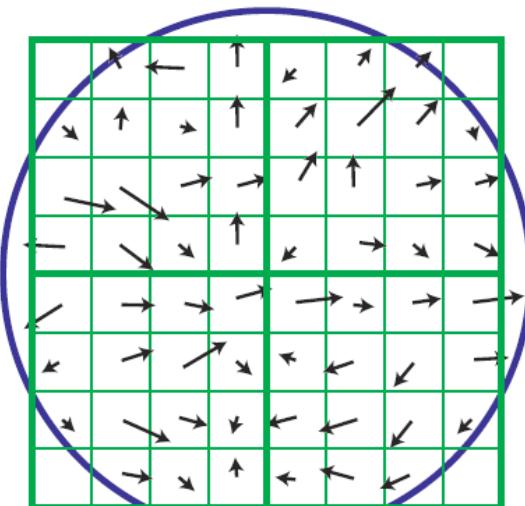
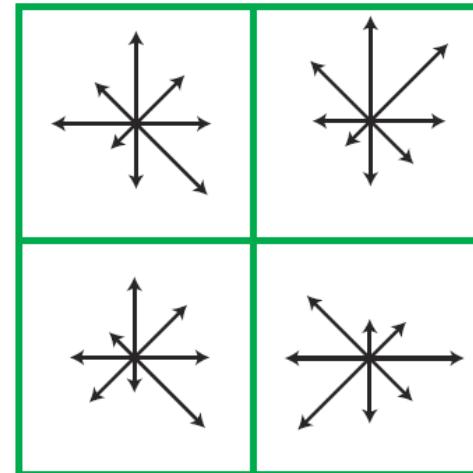
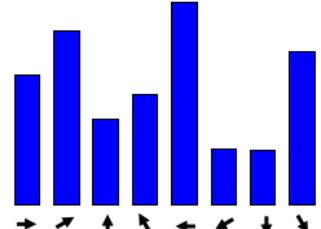


Image gradients



Keypoint descriptor

(la figura del paper muestra 2x2 zonas pero el descriptor usa 4x4 zonas)



# SIFT (D.Lowe)

- 16 histogramas con 8 orientaciones del gradiente=128 dimensiones
  - Cada gradiente suma en zonas y orientaciones vecinas (soft)
- Gaussiana entorno al keypoint (círculo azul) para suavizar valores y darle más importancia al centro
- Los gradientes son invariantes a cambios regulares de iluminación
- Vector normalizado a largo unitario: Invariante a cambio de contraste
- Cambios no regulares de la iluminación (ej: en objetos 3D) afectan más la magnitud de los gradientes que la orientación.
- Saturación del histograma: Cada valor superior a 0.2 es recortado y el vector es re-normalizado



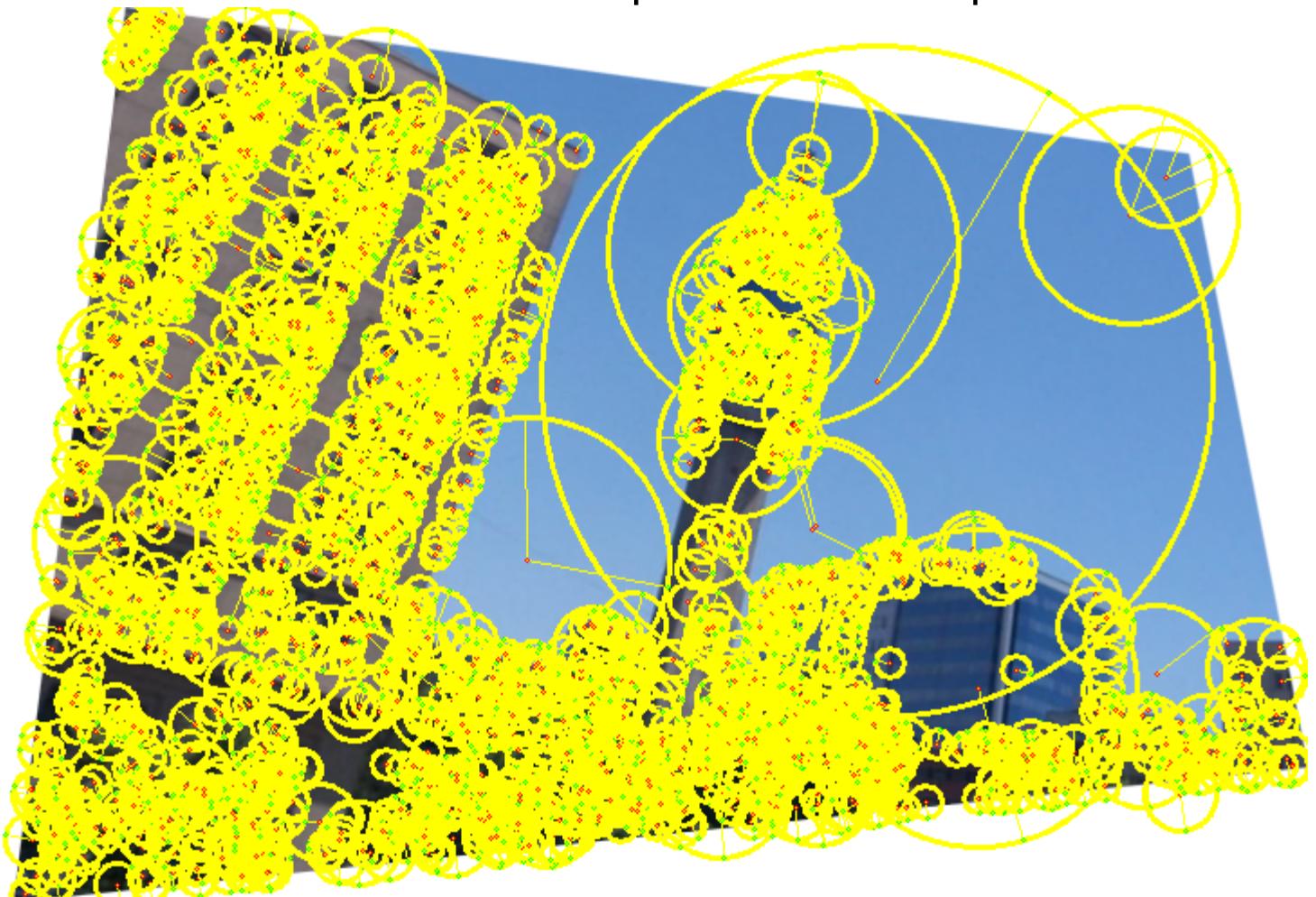
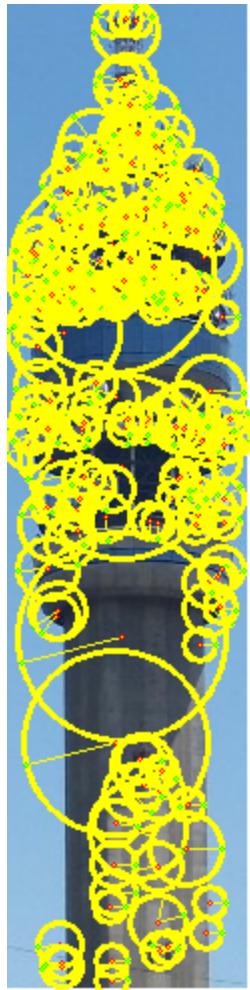
# **Coincidencias entre Imágenes**

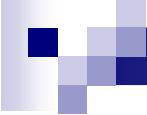
# Comparar descriptores locales



# Comparar descriptores locales

262 puntos vs 1996 puntos



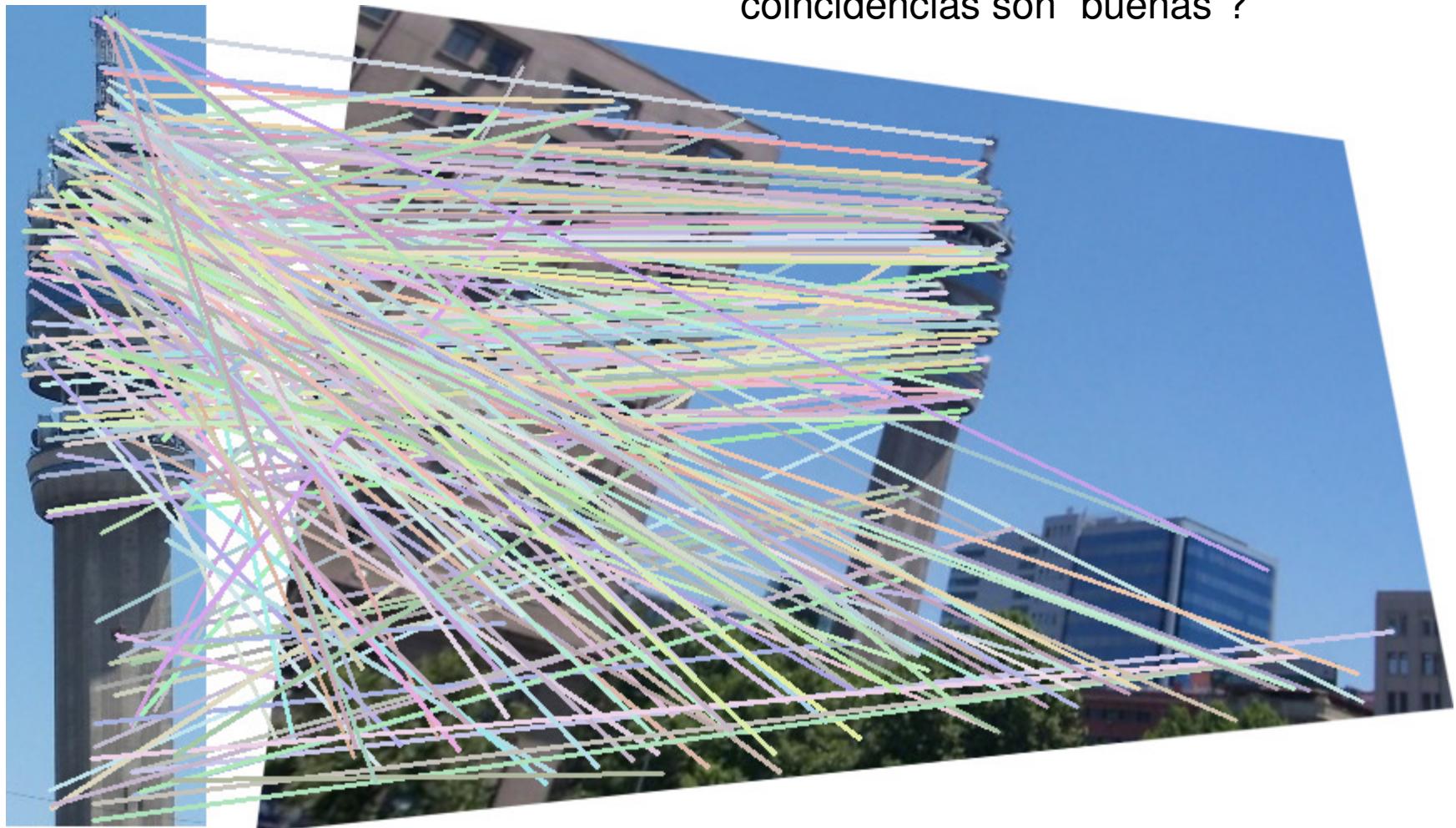


# Asociar descriptores similares

- Cada descriptor local de imagen 1 se compara con cada descriptor local de la imagen 2
  - Búsqueda kNN entre descriptores locales
  - Distancia Euclídea entre los descriptores locales (NO entre las coordenadas espaciales)
  - D.Lowe propone búsqueda aproximada usando Kd-Tree
- Las “coincidencias” entre dos imágenes son los pares  $(v_i, \text{1NN}(v_i))$  donde:
  - $v_i$  descriptor de imagen 1
  - $\text{1NN}(v_i)$  descriptor de imagen 2 con menor distancia a  $v_i$

# Asociar cada descriptor con su NN

¿Cómo determinar cuales de las 262 coincidencias son “buenas”?





# Seleccionar buenas coincidencias

## ■ Primer criterio:

- En algunos casos una distancia umbral fija no funciona bien.
- Un par  $(v_i, 1NN(v_i))$  es una “buena coincidencia” cuando su distancia es mucho menor a la distancia del segundo NN:  
$$\text{dist}(v_i, 1NN(v_i)) / \text{dist}(v_i, 2NN(v_i)) \leq \text{umbral}$$

D.Lowe propone umbral 0.8

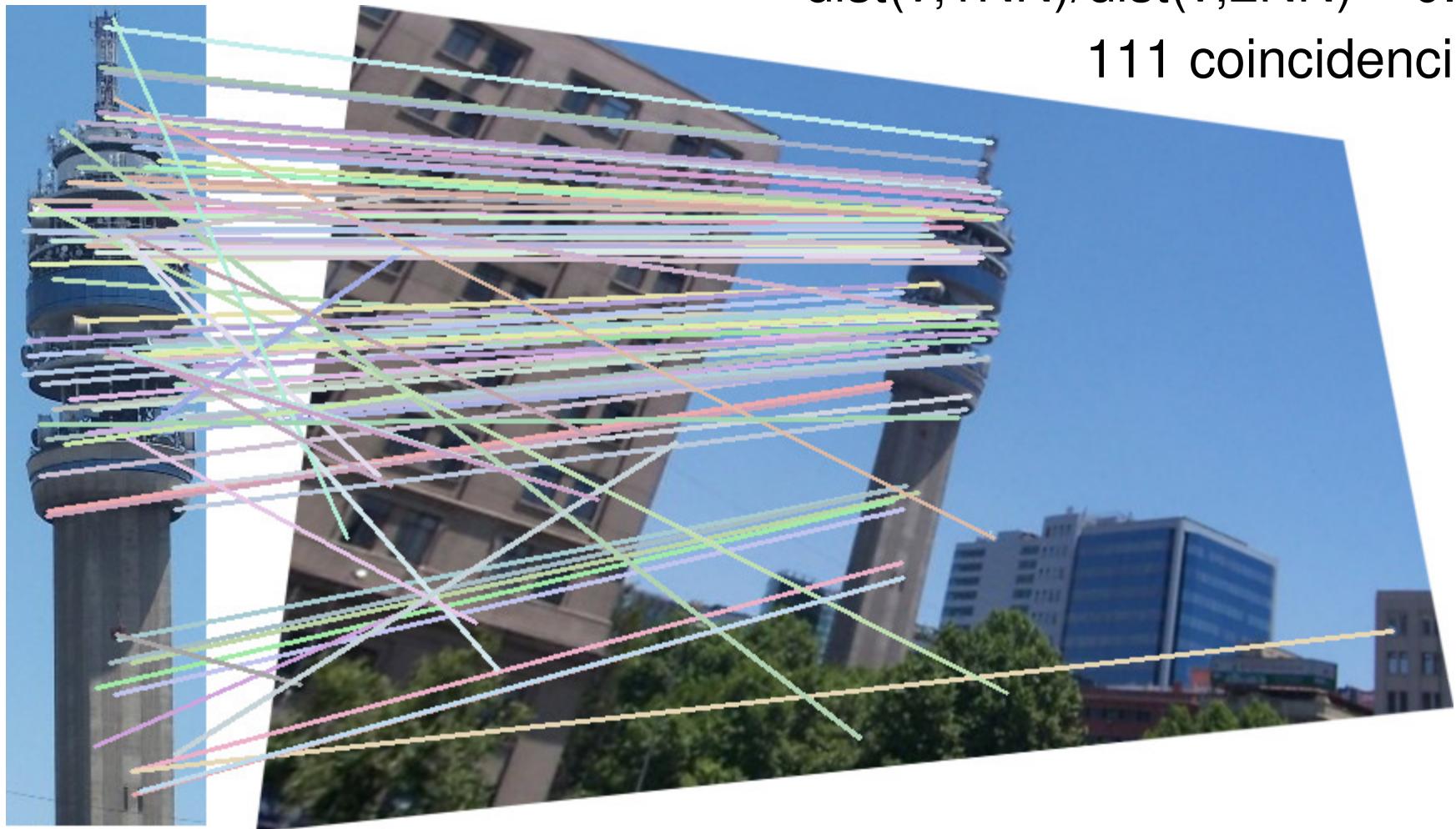
## ■ Segundo criterio: Coherencia Espacial

- Seleccionar grupos de coincidencias que apunten a un mismo lugar

# Seleccionar buenas coincidencias

$\text{dist}(v, \text{1NN})/\text{dist}(v, \text{2NN}) \leq 0.9$

111 coincidencias



# Seleccionar buenas coincidencias

$\text{dist}(v, \text{1NN})/\text{dist}(v, \text{2NN}) \leq 0.8$

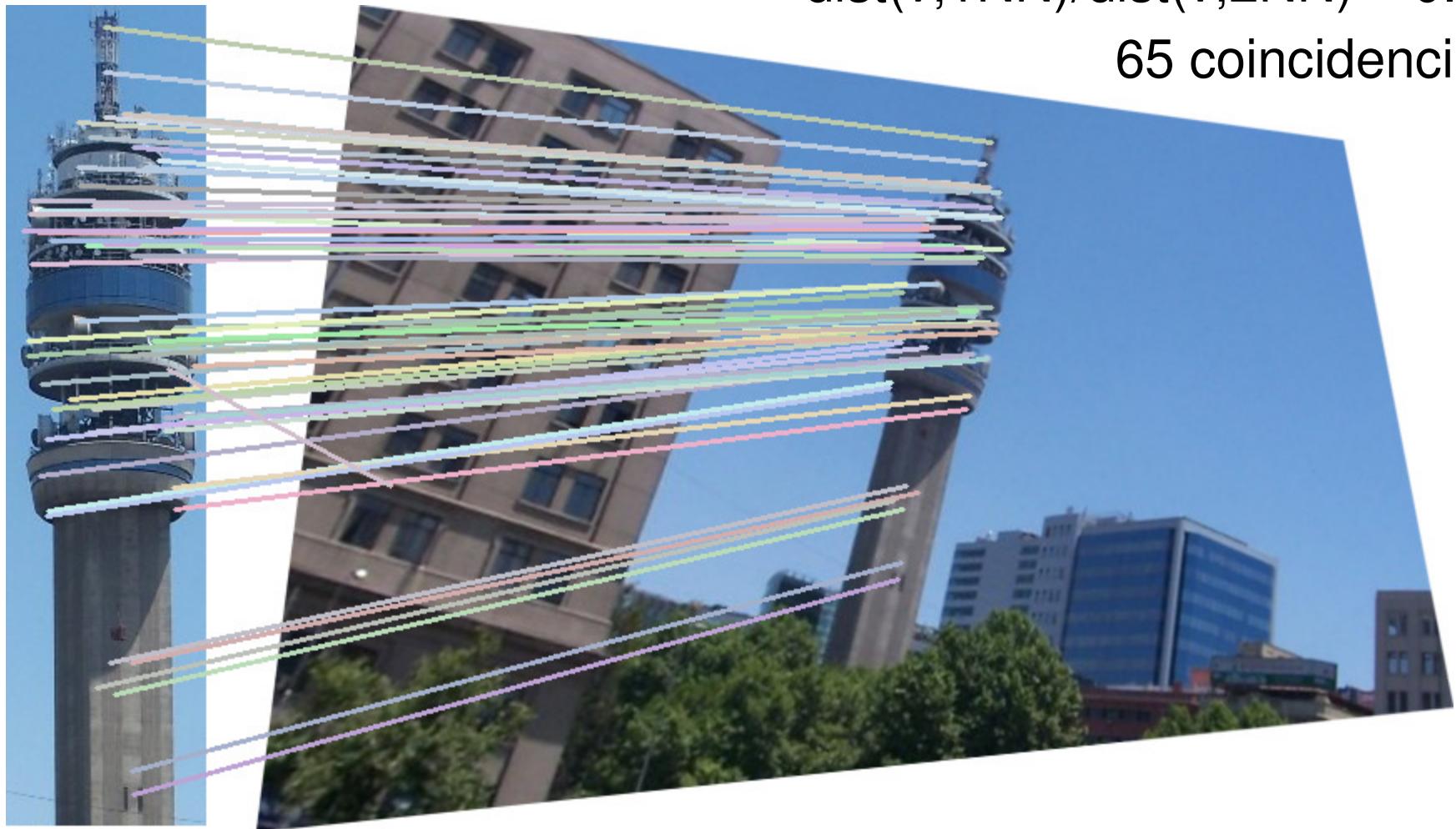
77 coincidencias



# Seleccionar buenas coincidencias

$\text{dist}(v, \text{1NN})/\text{dist}(v, \text{2NN}) \leq 0.7$

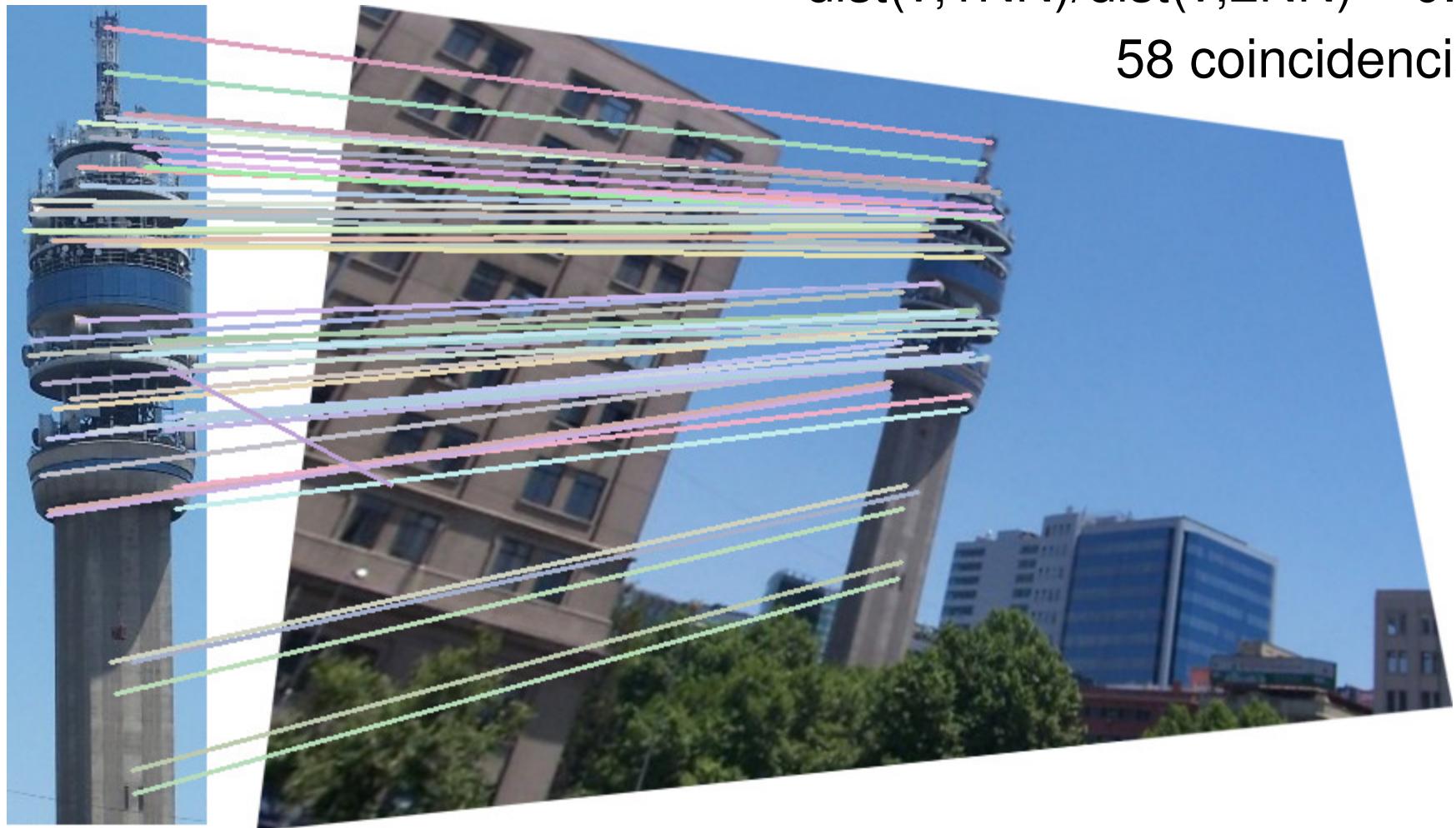
65 coincidencias



# Seleccionar buenas coincidencias

$\text{dist}(v, \text{1NN})/\text{dist}(v, \text{2NN}) \leq 0.6$

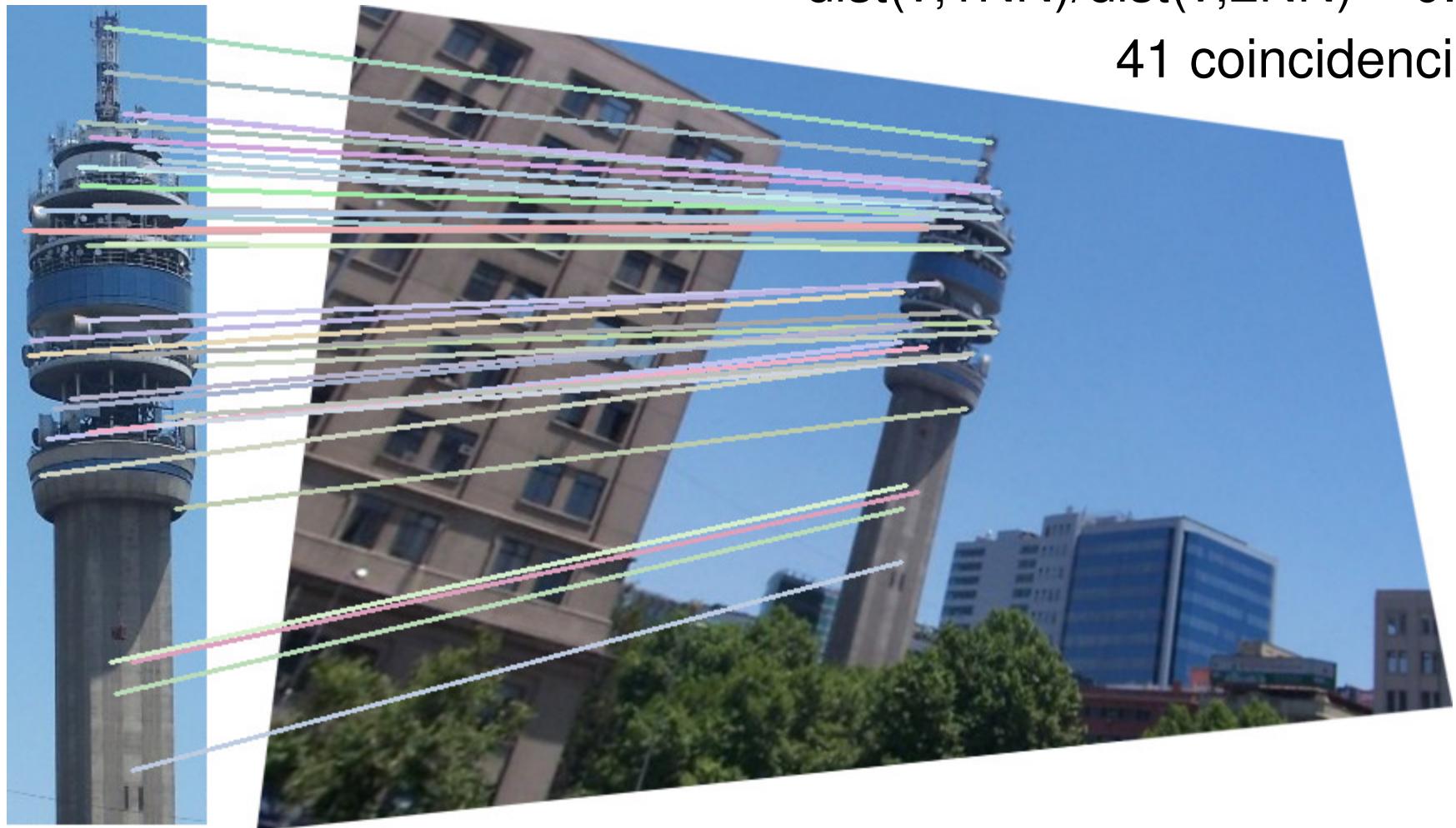
58 coincidencias



# Seleccionar buenas coincidencias

$\text{dist}(v, \text{1NN})/\text{dist}(v, \text{2NN}) \leq 0.5$

41 coincidencias



# Coherencia Espacial

- La coherencia espacial consiste en encontrar un patrón común entre las coordenadas espaciales  $(x,y)$  de los descriptores locales de dos imágenes
- Una transformación geométrica  $T$  calcula nuevas coordenadas espaciales  $T(x,y)=(x',y')$
- Dado un conjunto de coincidencias entre dos imágenes, cada una de la forma  $(P_x, P_y) \rightarrow (Q_x, Q_y)$
- Llamaremos “inliers” de  $T$  a las coincidencias que cumplen:  $T(P_x, P_y) \approx (Q_x, Q_y)$
- Deseamos encontrar la transformación  $T$  con la mayor cantidad de inliers posible

# Modelo de transformación

- Primero se debe escoger un modelo de transformación a buscar
  - Más restrictivas son más rápidas de encontrar
  - Más generales pueden encontrar objetos en distintas posiciones
- Si  $T$  sólo considera traslación  $(t_x, t_y)$  basta tomar una única coincidencia y  $T$  se define como:
  - $P_x + t_x = Q_x$
  - $P_y + t_y = Q_y$

$$\begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

# Modelo de transformación

- Si  $T$  es escala  $\sigma$  + traslación:

- $P_x * \sigma_x + t_x = Q_x$

- $P_y * \sigma_y + t_y = Q_y$

$$\begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

Requerirá de 2 coincidencias

(más simple:  $\sigma_x = \sigma_y$ )

- Si  $T$  es rotación  $\theta$  + escala  $\sigma$  + traslación (RST):

$$\begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

# Modelo de transformación

- Transformación Afín:

- Requiere 3 coincidencias

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

- Homografía (perspectiva):

- Requiere 4 coincidencias

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix} = \begin{bmatrix} wQ_x \\ wQ_y \\ w \end{bmatrix}$$

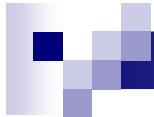


Image 1



Image 2



Rotation, Scale and Translation



Affine



Homography



# Encontrar Transformación

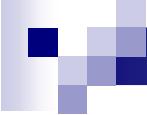
- El tipo de transformación define los grados de libertad que se deben fijar
  - Usualmente  $T$  requiere fijar 2 a 8 parámetros (1 a 4 coincidencias entre imágenes)
- Algoritmos para encontrar la mejor transformación  $T$  y el conjunto de inliers  $S$ :
  - RANSAC (Random Sample Consensus)
  - Transformada de Hough

# RANSAC

- Método aleatorio que genera posibles modelos, los evalúa y selecciona el mejor
- Algoritmo RANSAC:
  1. Realizar  $N$  veces:
    - a) Seleccionar al azar las coincidencias necesarias para definir  $T'$  (semillas)
    - b) Buscar todos las coincidencias que cumplen el modelo  $T'$  (inliers) según una tolerancia  $\epsilon$  ( $S'$ )
    - c) Opcional, corregir el modelo usando mínimos cuadrados entre los inliers
    - d) Guardar  $T'$  y  $S'$  si la cantidad de inliers encontrados supera al mejor modelo encontrado  $T$  y  $S$
  2. Seleccionar el modelo  $T$  que obtuvo más inliers  $S$

# Probabilidad de éxito RANSAC

- Ejemplo, se desea buscar de una homografía entre dos imágenes usando RANSAC
- Se tienen 1000 coincidencias iniciales
- Si  $T$  tiene 20 inliers la probabilidad de seleccionar cuatro inliers al azar:
  - $P_{sel} = 20/1000 * 19/999 * 18/998 * 17/997 \approx 10^{-7}$
- Probabilidad de encontrar la homografía correcta luego de  $N$  intentos:
  - $P_{exito} = 1 - (1 - P_{sel})^N$
- Luego de 6.000.000 intentos  $P_{exito} \sim 50\%$



# Transformada de Hough

- Método exhaustivo que vota por todas las  $T$  posibles para cada coincidencia y selecciona la más votada
- Espacio de los parámetros: las dimensiones son los grados de libertad del modelo a buscar
  - Definido por el tipo de transformación
  - Cada coincidencia vota por todas las transformaciones de las que puede ser parte
  - Se busca el conjunto de parámetros más votado



# Transformada de Hough

## ■ Algoritmo T. Hough:

- Dividir el espacio de los parámetros en celdas de votación (discretizar cada dimensión en un número fijo de rangos)
- Cada celda es una posible Transformación
- Para cada coincidencia  $P \rightarrow Q$ :
  - Sumar 1 a todas las transformaciones que convierten las coordenadas de P en las coordenadas de Q
- Seleccionar la transformación correspondiente a la celda con mayor votación

# Transformada de Hough

- Voto ponderado entre celdas cercanas
- Una matriz dispersa para guardar sólo los contadores mayores a 0
- Modelos con muchos grados de libertad afecta la eficiencia
- Además de la ubicación espacial, usar escala y orientación de los puntos de interés:
  - $(P_x, P_y, P_\sigma, P_\theta) \rightarrow (Q_x, Q_y, Q_\sigma, Q_\theta)$
  - Transformación RST definida por una única coincidencia



# Resumen

## ■ RANSAC:

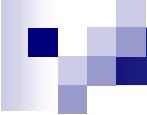
- Debe ser probable encontrar un modelo correcto a partir de selección al azar (deben existir muchos inliers)

## ■ Transformada de Hough:

- Puede encontrar más de una transformación
- Funciona independiente de la cantidad de outliers

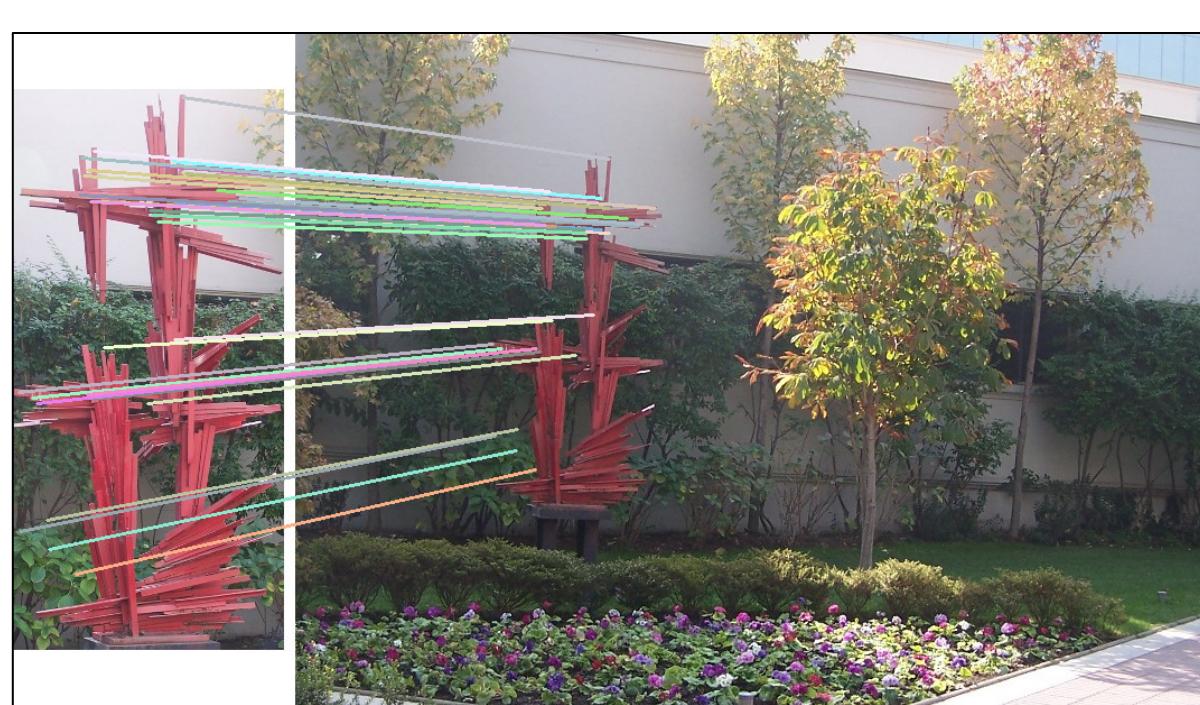
# Comparación entre imágenes

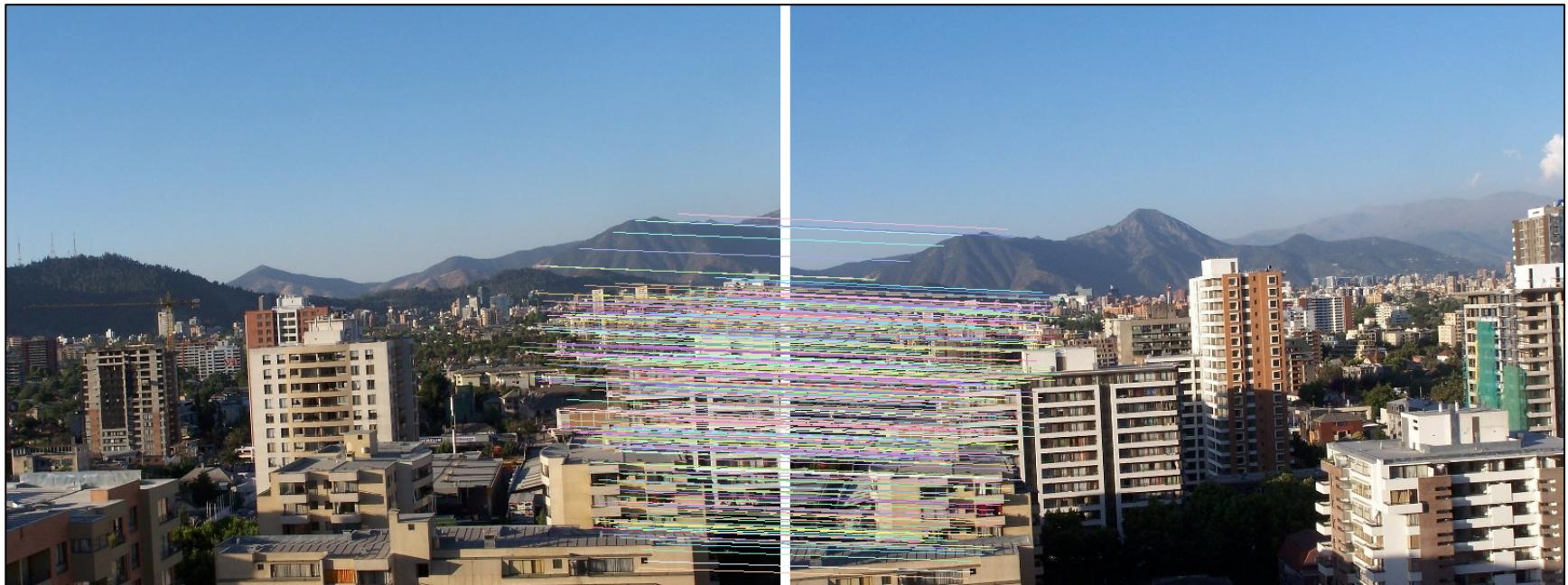
- Si  $q$  es imagen de consulta,  $n_q$  es el número de descriptores locales
  - similitud( $q,r$ ) = inliers( $T$ )
  - distancia( $q,r$ ) =  $1 - \text{inliers}(T) / n_q$ 
    - Pero no es métrica
- Comparar cada descriptor de  $q$  con los descriptores de todas las imágenes
  - Una colección de imágenes puede contener del orden de cientos de millones de descriptores locales
  - Requiere indexamiento y búsqueda aproximada



# Ejemplos

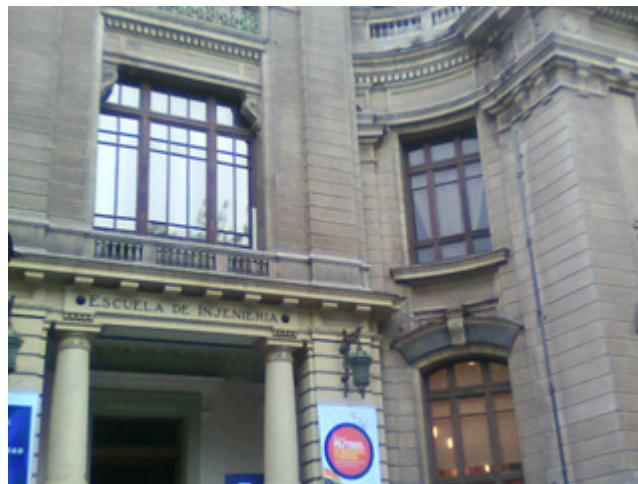
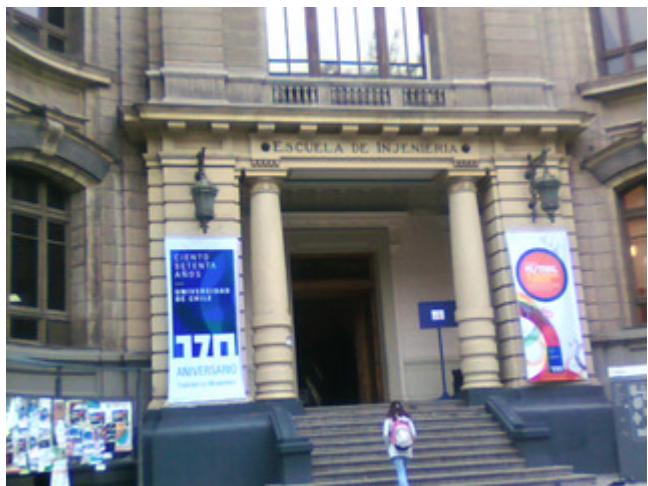
- Descriptores locales se pueden usar para:
  - Búsqueda de objetos
  - Panorámicas (image stitching)
  - Clasificación de imágenes
  - Realidad aumentada
  - Clasificación de imágenes



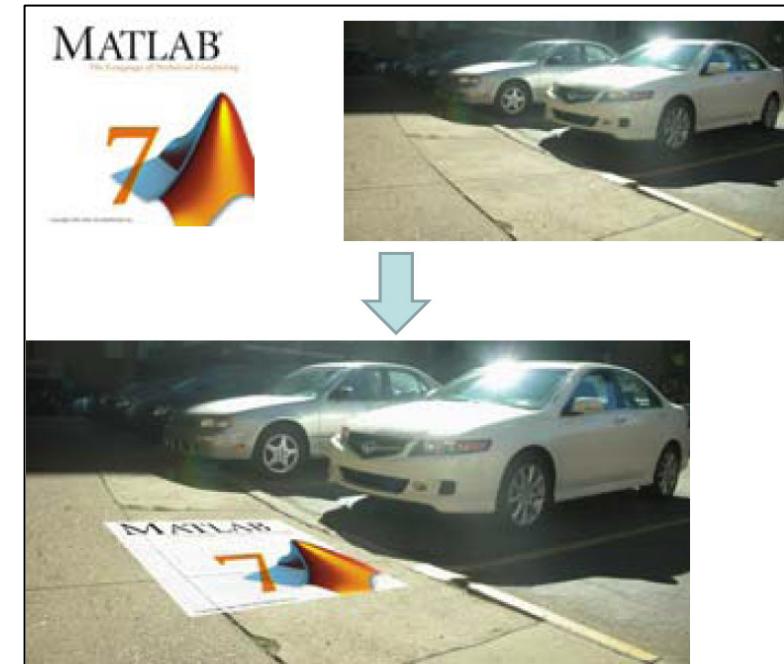






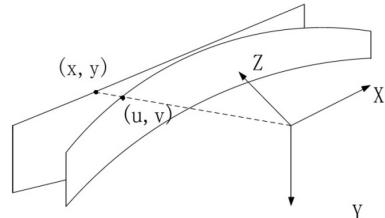




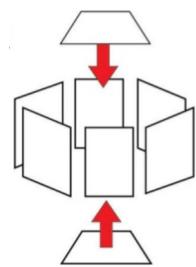


# Image Stitching

- Una homografía puede no ser lo más adecuado para unir imágenes y construir panorámicas
- Usar otras transformaciones como:
  - Transformaciones Cilíndricas:



- Transformaciones Esféricas:





# Otros detectores de puntos de interés

- **MSER.** Detector de zonas estables.
- **Hessian-Laplace.** Similar a Harris pero usando segundas derivadas.
- **FAST.** Detector de esquinas.

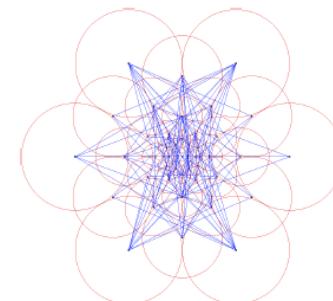
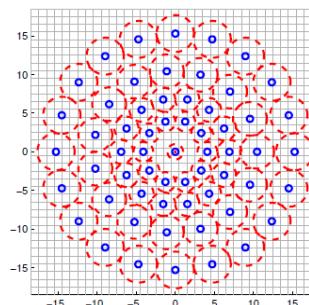
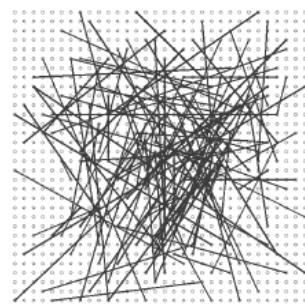


# Otros descriptores locales

- **SURF.** Aproximación a SIFT más rápida de calcular, define un detector y un descriptor
- **GLOH.** Descriptor similar a SIFT con más regiones y PCA
- **DAISY.** Se utilizan regiones circulares solapadas.
- **RootSIFT.** Cambiar la normalización y utilizar otra distancia
- Considerar colores: **CSIFT**, **OpponentSIFT**, **HueSIFT**

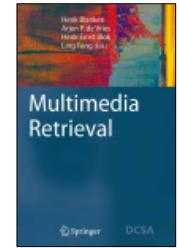
# Descriptores Binarios

- Secuencia de bits para representar una zona
  - Se compara la intensidad de dos zonas. Si hay aumento es 1 y 0 si no.
  - Se comparan varios pares de zonas (ej. 512 pares → 512 bits)
- Descriptoros se comparan con distancia de Hamming
- Por ejemplo: BRIEF, ORB, BRISK, FREAK, LATCH



# Bibliografía

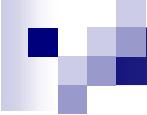
- **Computer Vision: Algorithms and Applications.** Szeliski, 2011.
  - Sec 4.1
- **Multimedia Retrieval.** Blanken et al. 2007.
  - Sec 5.4.2 (Harris)
- **Distinctive image features from scale-invariant keypoints.** Lowe, 2004.





# Papers puntos de interés

- Harris and Stephens. “A combined corner and edge detector”. 1988.
- Shi and Tomasi. “Good features to track”. 1994.
- Lowe. “Distinctive image features from scale-invariant keypoints”. 2004.
- Matas et al. “Robust Wide Baseline Stereo from Maximally Stable Extremal Regions”. 2002.
- Mikolajczyk et al. “Scale & Affine Invariant Interest Point Detectors”. 2004.
- Rosten et al. “Machine learning for high-speed corner detection”. 2006.
- Rosten et al. “Faster and Better: A Machine Learning Approach to Corner Detection”. 2010.



# Papers descriptores locales

- Lowe. “Distinctive image features from scale-invariant keypoints”. 2004.
- Mikolajczyk et al. “Performance Evaluation of Local Descriptors”. 2005.
- Arandjelovic et al. “Three things everyone should know to improve object retrieval”. 2012.
- Bay et al. “Speeded-Up Robust Features (SURF)”. 2008.
- Tola et al. “DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo”. 2010.
- Abdel-Hakim et al. “CSIFT: A SIFT Descriptor with Color Invariant Characteristics”. 2006.
- Van de Sande et al. “Evaluation of Color Descriptors for Object and Scene Recognition”. 2008.
- Mazin et al. “Combining color and geometry for local image matching”. 2012.



# Papers descriptores binarios

- Calonder et al. “Brief: Binary robust independent elementary features”. ECCV, 2010.
- Rublee, Ethan, et al. “ORB: an efficient alternative to SIFT or SURF”. ICCV, 2011.
- Leutenegger et al. “BRISK: Binary robust invariant scalable keypoints”. ICCV, 2011.
- Alahi et al. “Freak: Fast retina keypoint”. CVPR, 2012.
- G. Levi and T. Hassner, “LATCH: Learned Arrangements of Three Patch Codes”. WACV, 2016.



# Implementaciones

- David Lowe  
<http://www.cs.ubc.ca/~lowe/keypoints/>
- VLFeat (Andrea Vedaldi)  
<http://www.vlfeat.org/>
- Krystian Mikolajczyk  
<http://kahlan.eps.surrey.ac.uk/featurespace/web/>
- Koen van de Sande  
<http://koen.me/research/colordescriptors/>
- OpenCV (SIFT, SURF, DAISY, binarios)
  - features2d, xfeatures2d (opencv\_contrib)