Visión por Computadora I

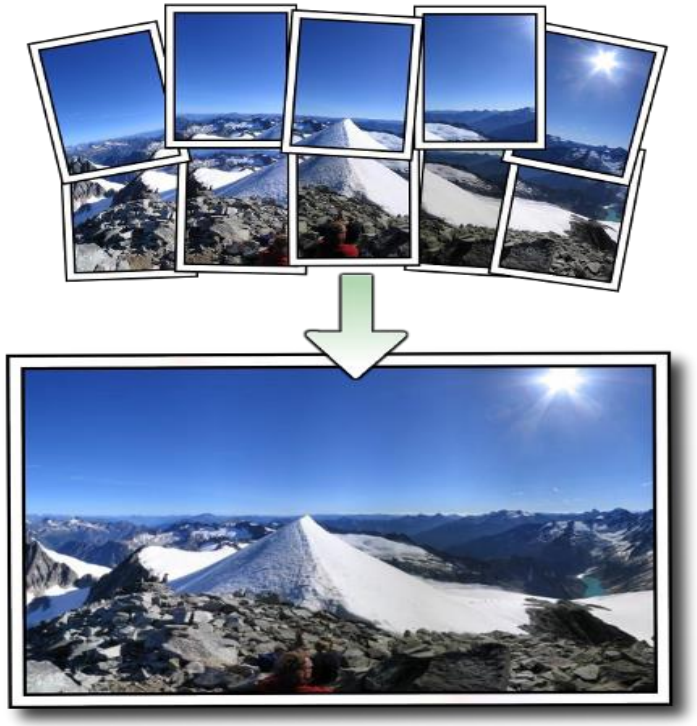
Ing. Maxim Dorogov

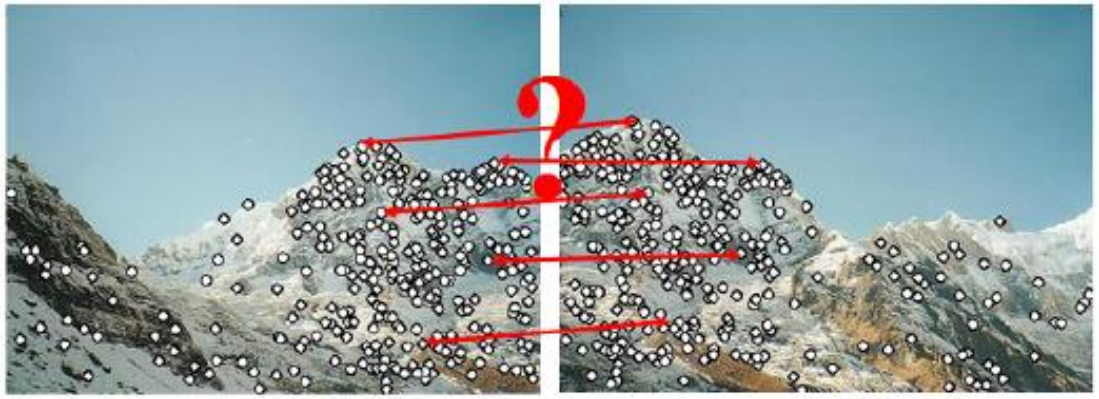
(mdorogov@fi.uba.ar)

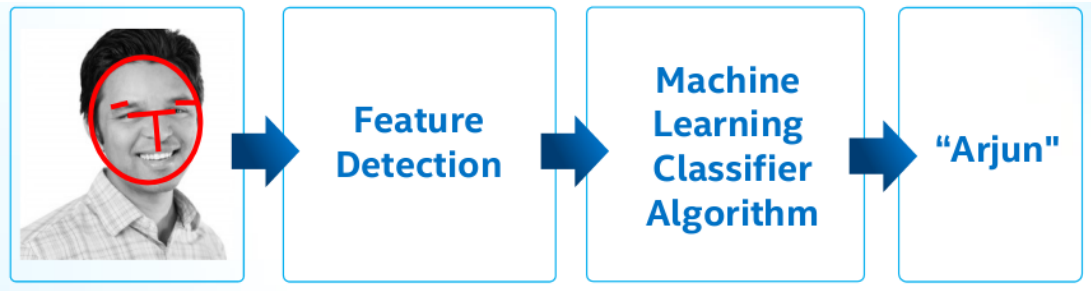
Laboratorio de Sistemas Embebidos -FIUBA

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Descriptores: Motivación

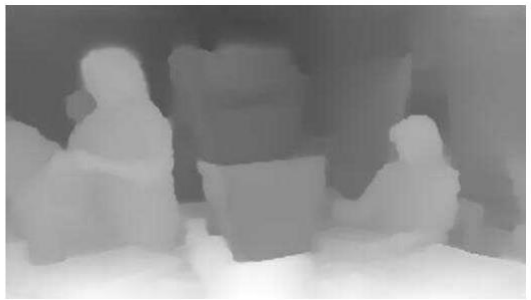






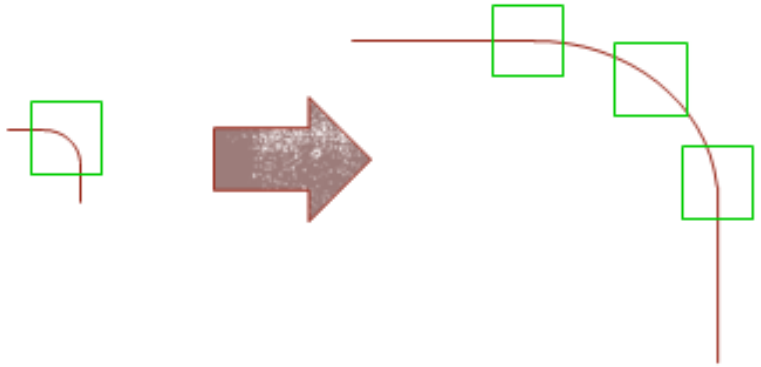
Algunas aplicaciones:

1. Extracción de características
2. Correspondencias estéreo
3. Structure from motion
4. Reconstrucción 3D
5. Clasificación de objetos
6. Reconocimiento facial



SIFT (SCALE INVARIANT FEATURE TRANSFORM)

- Las esquinas, por supuesto, son invariantes a la rotación (una esquina sigue siendo una esquina aún rotada), pero ¿qué pasa si cambiamos la escala?
- Una esquina puede dejar de ser una esquina dependiendo la escala. Es decir, a veces no nos conviene buscar características a la mínima escala posible (por ejemplo cuando se trabaja con imágenes de baja frecuencia espacial, como nubes).
- D. Lowe (2004) publica "Distinctive Image Features from Scale-Invariant Keypoints" donde plantea un algoritmo (SIFT) para extraer características y computar sus descriptores.
- Hay básicamente 4 pasos en el algoritmo propuesto:
  1. **Construcción de un espacio de escalas y detección de extremos.**
     - LoG ó DoG (invariantes a escala y rotación)
  2. **Localización de puntos clave**
     - Encontrar la escala y ubicación de los puntos candidatos
  3. **Asignación de orientación**
     - Asignación de una o más orientaciones a cada punto clave
  4. **Descriptor de puntos clave**
     - Con los gradientes (orientaciones) y ubicación de los puntos clave se genera un descriptor que también es invariante a distorsiones geométricas locales y cambios de iluminación.

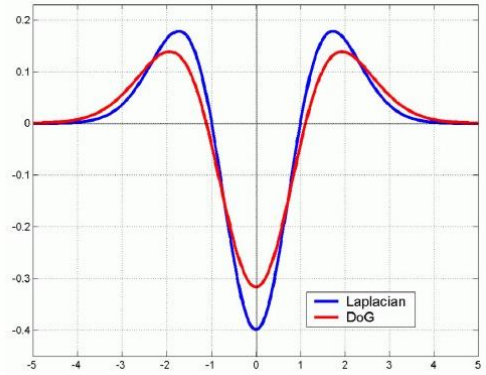


$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

Laplaciano de Gaussiana

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

Diferencia de Gaussianas



SIFT

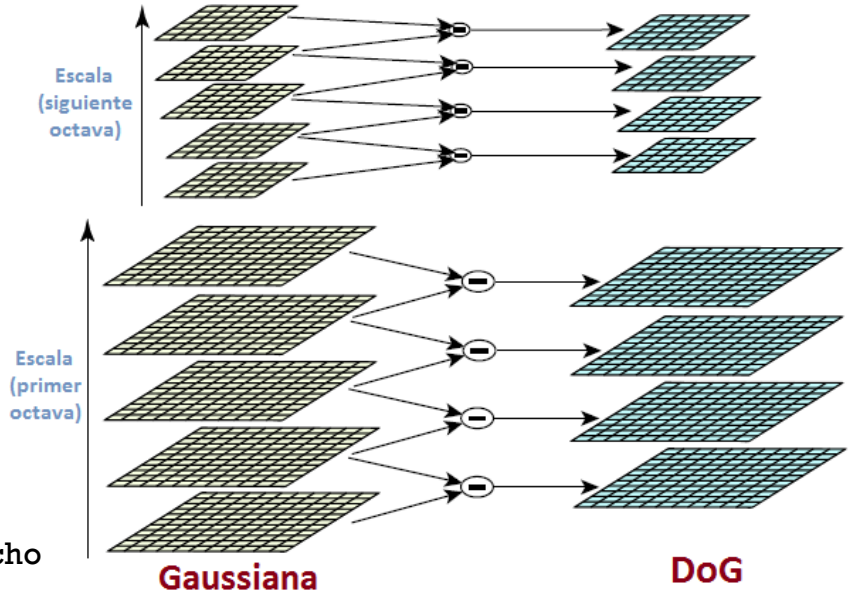
Detección de extremos

- Los máximos y mínimos de $\sigma^2\nabla^2 G$ producen características mas estables que otras funciones tales como el gradiente, Hessiano o Harris. (Mikolajczyk, 2002)
- Se aproxima LoG con DoG (computacionalmente mas eficiente) como:

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G.$$

- Así, mientras el factor (k - 1) se mantenga constante se incorpora el factor de normalización σ^2 necesario para que el Laplaciano sea invariante por la escala.
- Cada octava (doble σ) se divide en un numero entero de escalas S, tal que $k = 2^{1/s}$. Se deben producir $s + 3$ imágenes borrosas para cada octava.



Localización de puntos clave

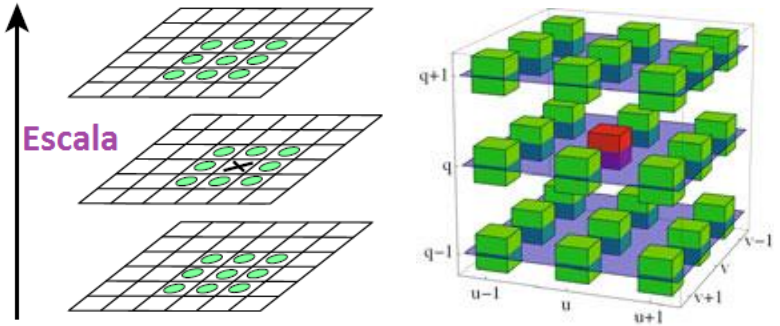
- Se buscan extremos locales (máx., y min. en D(x, y, **σ**)). Para esto se compara cada píxel con sus ocho vecinos en la imagen actual y los nueve vecinos en las dos capas adyacentes.
- Se interpola haciendo un desarrollo de Taylor (orden 2) en torno al extremo, se deriva para buscar un extremo local y se despeja un **δX** = (δx,δy,δσ). Si **δX** es > 0.5 en cualquiera de sus coordenadas se descarta el extremo, caso contrario se suma **δX** al extremo original.

Se deben eliminar:

- Puntos de bajo contraste (más susceptibles a ruido). Brown y Lowe, 2002.

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}.$$ *Se elimina todo extremo con |D(x)| < 0.03*

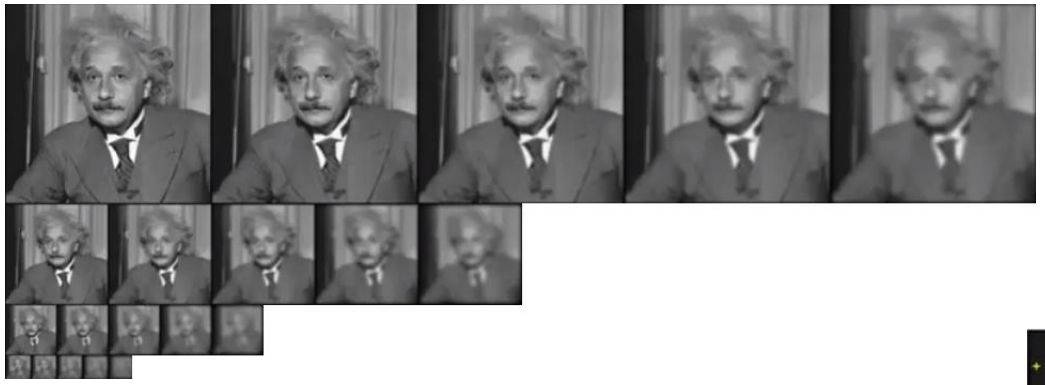
- Puntos que corresponden a bordes debido a que no suelen aportar información relevante.

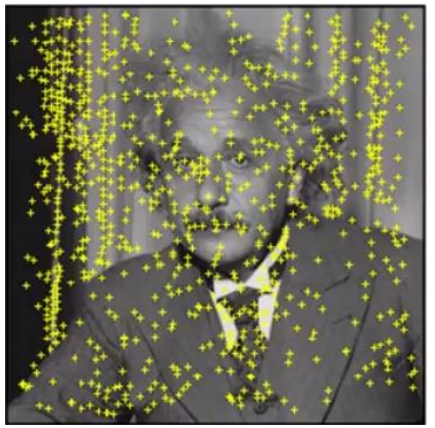


$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \qquad \frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$ r = 10 (relación entre el autovalor más grande y más chico)

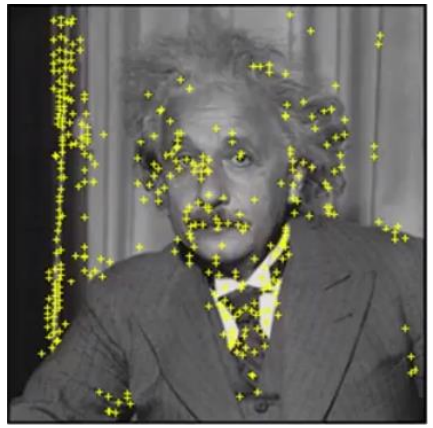
SIFT – EJ: LOCALIZACIÓN

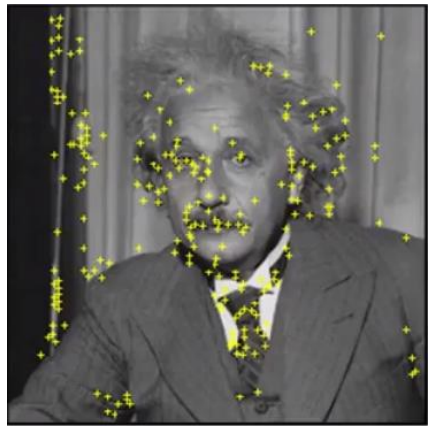
Contraste > C y $\Delta x < 0.5$





Máximos locales





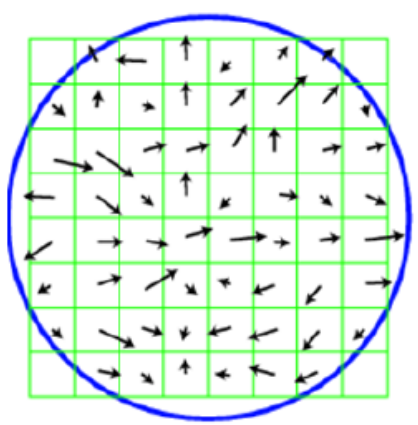
Quitar bordes

SIFT

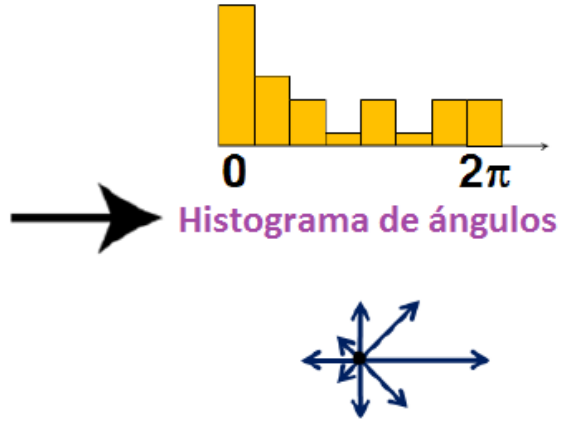
3. **Asignación de orientación**
   - Una vez detectado el punto clave se busca (dentro de una ventana proporcional a la escala) la magnitud y gradiente en esa region, en su escala. Para eso realiza un histograma de 36 bins cubriendo los 360°. Las magnitudes están pesadas por una Ventana circular gaussiana de $\sigma = 1.5$ la escala.
   - Se elige el valor del histograma más alto y todo valor dentro del 80% de este se utiliza para generar nuevos puntos claves en el mismo espacio-escala pero con diferentes orientaciones.

4. **Descriptor de puntos claves**
   - Una vez determinada la posición, escala y orientación principal se genera el descriptor para poder realizar la correspondencia en otra imagen.
   - Para eso se toma un entorno de 16x16 vecinos alrededor del punto clave (sin rotar).
   - Este entorno es dividido en bloques de 4x4 sobre los que se calcula un histograma de orientaciones con 8 bins cada uno. Se utiliza una ventana circular gaussiana de σ igual a la mitad del ancho de la ventana del descriptor (robustez frente a desplazamiento de la ventana y minimizar gradientes lejanos).
   - Para lograr la invariancia a rotación las coordenadas del descriptor y las orientaciones de los gradientes son rotadas relativas a la rotación principal del punto clave.
   - Así se obtiene un descriptor por punto clave como un vector de 128 elementos.
   - Este vector se normaliza a longitud unitaria de manera de ser invariantes a cambios de contraste (el descriptor per se es invariante a cambios de brillo)
   - Los gradientes de gran magnitud suelen dominar en el armado de histogramas, por lo que se recomienda recortar todos las magnitudes que superen 0,2 y luego renormalizar

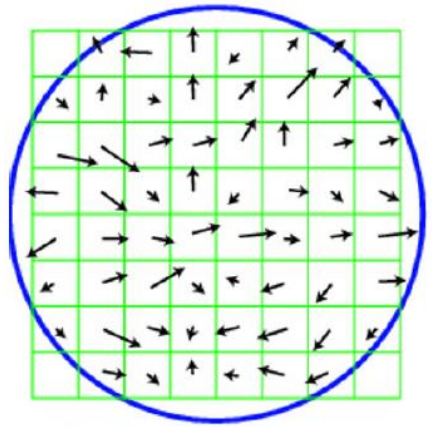
5. **Correspondencia de puntos claves**
   - Los puntos claves entre dos imágenes se corresponden buscando los vecinos más cercanos. En algunos casos la segunda correspondencia más cercana está cerca también. Si la relación de distancias entre la primer correspondencia y la segunda es mayor a 0,8 se descartan (las dos). Esto elimina el 90% de las falsas correspondencias y el 5% de las correctas.
   - Los descriptores pueden comprimirse usando distintos métodos como PCA, LDA, etc. Inclusive utilizar LSH (Local Sensitive Hashing) para pasar de datos float a datos binarios. Luego estos binarios pueden compararse usando distancia de Hamming.



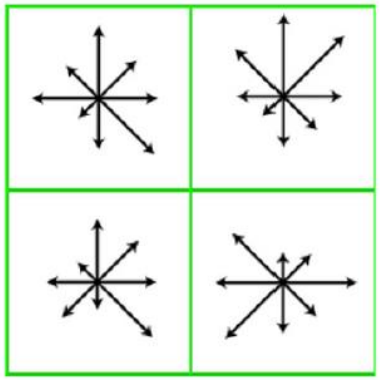
Gradientes de la imagen



Histograma de ángulos



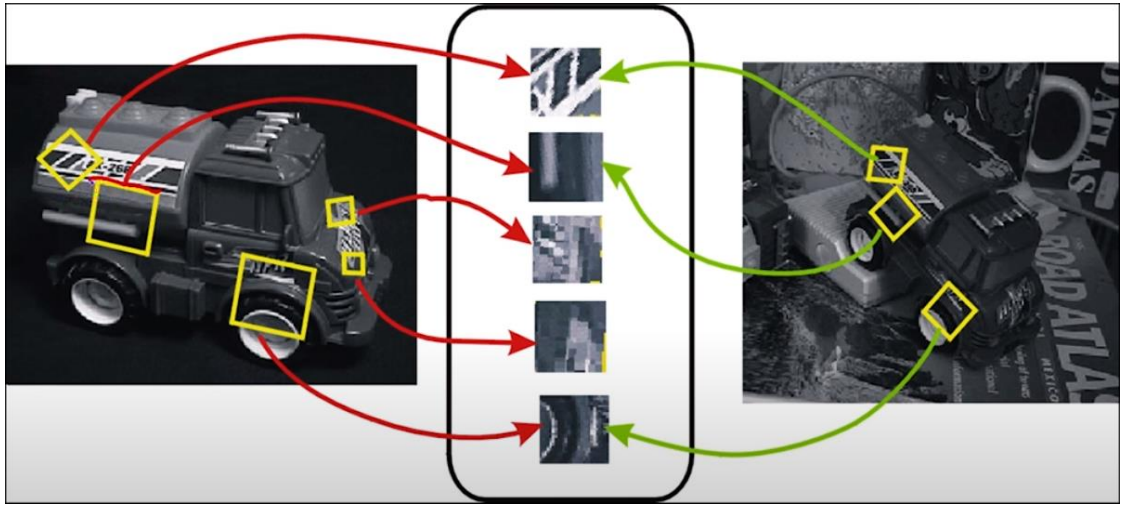
Gradientes en la imagen



Descriptor de punto clave

SIFT

Puntos clave en escala, espacio, orientación

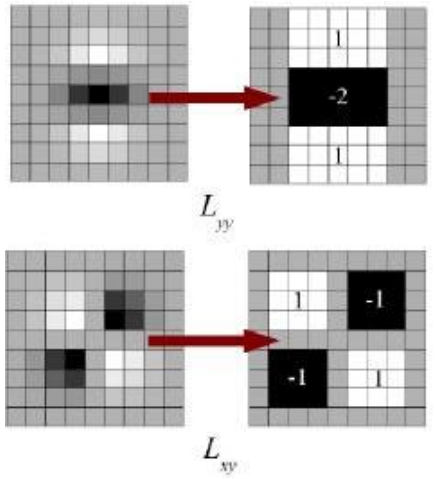


Correspondencia de puntos clave

Invariante a la escala

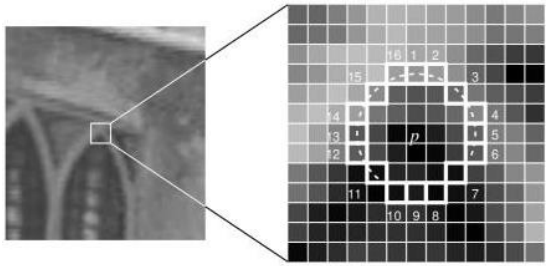
VARIANTES A SIFT

1. **SURF (Speeded-Up Robust Features)**
   - Variante más rápida a SIFT introducida por Bay, H., Tuytelaars, T. and Van Gool, L (2006)
   - Aproxima la LoG con un filtro cuadrado que tiene la ventaja de poder calcularse fácilmente con imágenes integrales. Además se puede procesar en paralelo a distintas escalas
   - Para la asignación de la orientación usa wavelets verticales y horizontales (también con pesos gaussianos). La respuesta wavelet también se puede calcular usando imágenes integrales
   - El descriptor SURF tiene 64 dimensiones, aunque existe una versión extendida de 128 para mayor discriminación



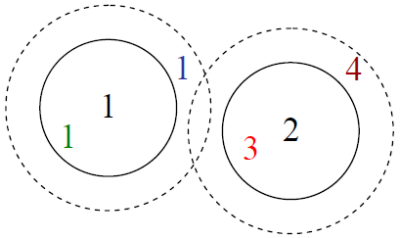
2. **FAST (Features from Accelerated Segment Test)**
   - Pensado para aplicaciones en tiempo real como SLAM
   - Propuesto por Edward Rosten y Tom Drummond en 2006 (revisado en 2010)
   - Dado un pixel p en la imagen se considera un círculo de 16 pixels alrededor. Luego, este pixel será una esquina si de estos 16 pixels hay **n** pixels contíguos que sean más brillantes o más oscuros que el valor de intensidad de **p** más un umbral **t** predefinido (n = 12 por defecto)
   - Corre un test rápido inicial solo con 4 vecinos (para descartar la mayoría de los no-esquinas), los que sobreviven corren el test de 16 vecinos
   - Finalmente corre una supresión de no-máximos para eliminar los puntos clave adyacentes
   - Es varias veces más rápido que otros detectores de esquinas no es robusto frente a altos niveles de ruido. Además depende de un umbral predefinido. Además no es invariante a rotación.

3. **BRIEF**
   - No es un método de detección de características sino un algoritmo de generación de descriptores
   - SIFT usa descriptores float de 128 dimensiones → 512 bytes por descriptor (SURF también, 256 bytes)
   - Toma un parche suavizado alrededor del punto clave y elige $n_d(x, y)$ pares de ubicaciones en un modo único (descrito en el paper). Luego compara la intensidad en cada par, por ejemplo p y q y si $I(p) < I(q)$ asigna un 1, si no un 0. n_d puede ser 128, 256 o 512 (16 bytes, 32 bytes o 64 bytes)



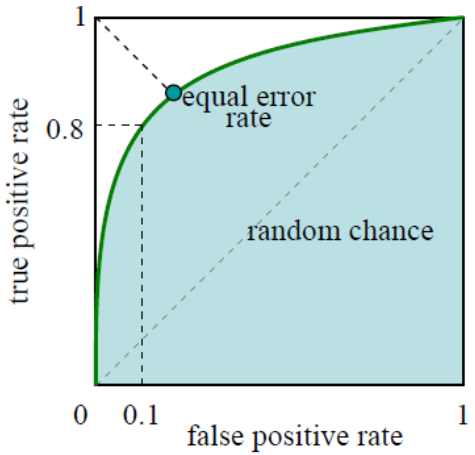
4. **ORB (Oriented FAST and Rotated BRIEF)**
   - SIFT y SURF…están patentados, por lo que hay que pagar para utilizarlos!
   - ORB es de OpenCV Labs y fue creado por Ethan Rublee, Vincent Rabaud, Kurt Konolige y Gary R. Bradski, 2011
   - Es una fusión entre FAST para detección de puntos clave y BRIEF para los descriptores
   - Para resolver el problema de la varianza a la rotación de FAST calcula el centroide de intensidad en cada parche. La dirección del vector del punto clave al centroide da la orientación
   - Los descriptores resultantes de BRIEF también incluyen esta orientación para robustecerse frente a rotaciones.

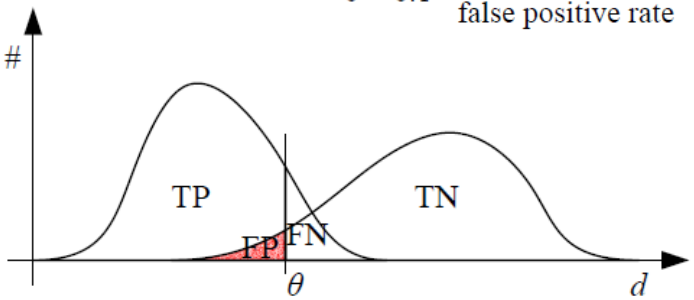
CORRESPONDENCIA DE CARACTERÍSTICAS



TP = 18	FP = 4	P' = 22
FN = 2	TN = 76	N' = 78
P = 20	N = 80	Total = 100

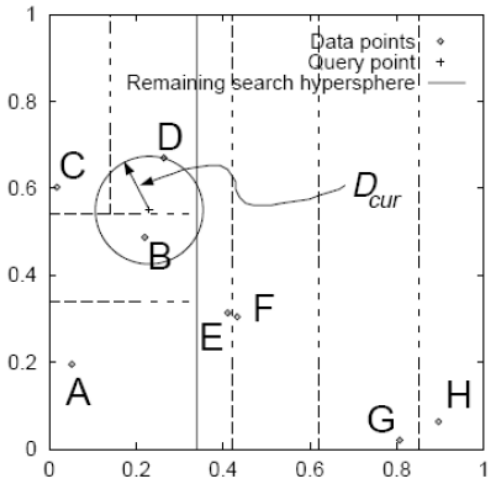
- $\boldsymbol{TPR} = \frac{TP}{TP+FN} = \frac{TP}{P}$
- $\boldsymbol{FPR} = \frac{FP}{FP+TN} = \frac{FP}{N}$
- $\boldsymbol{PPV} = \frac{TP}{TP+FP} = \frac{TP}{P'}$
- $\boldsymbol{ACC} = \frac{TP+TN}{P+N}$

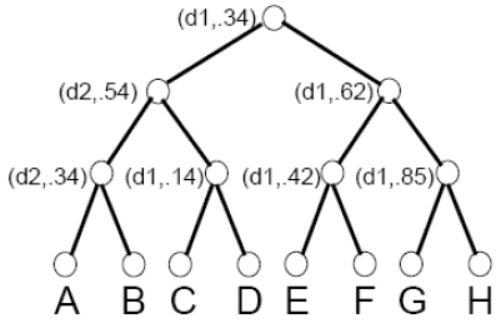


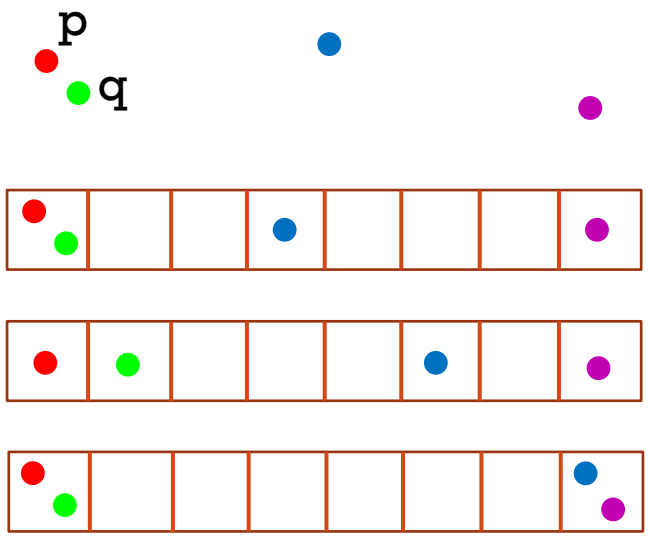


- Una vez obtenidos los puntos clave (keypoints) en distintas imágenes surge la pregunta de cómo hacer las correspondencias.
  - Para compararlos en general habrá que definir alguna medida de distancia:
    - Norma 1 (L1): $\|\dot{x}\|_1 = |x_1| + |x_2| + |x_3| \ldots + |x_n|$
    - Norma 2 (L2): $\|\dot{x}\|_2 = \sqrt[2]{|x_1|^2 + |x_2|^2 + |x_3|^2 + \cdots + |x_n|^2}$
    - Norma ∞ (INF): $\|\dot{x}\|_\infty = máx(|x_1|, |x_2|, |x_3| \ldots, |x_n|)$
    - Hamming (para códigos binarios): Cantidad de bits que deben cambiarse para trasformar un código en el otro. Se puede calcular como la suma de bits '1' luego de aplicar una función XOR entre los dos códigos. Ej: $1011011 \text{ XOR } 1101001 = 0110010 \rightarrow d = 3$
  - Existe también un proceso llamado *whitening* que involucra transformar los descriptores a otra base escalada y que se utiliza en el contexto de reconocimiento de rostros (eigen-faces)
- ¿Cómo comparamos las distancias de los keypoints en cada imagen?
  1. **Fuerza bruta**: Comparar cada uno de los keypoints en una imagen contra los de la otra
  2. **Diseñar una estructura de indexación**: Hay dos posibilidades
     1. Definir un árbol de búsqueda multidimensional (k-D tree)
     2. Definir una hash table (locality sensitive hashing)

CORRESPONDENCIA DE CARACTERÍSTICAS







- **k-D Tree**
  - SIFT usa BBF (Best Bin First) que es una modificación al algoritmo k-d Tree típico (Beis & Lowe, 1997)
  - Permite acceder a keypoints a razón de $\log(n)$ en lugar de proporcional a n.
  - Puede acelerar el proceso de búsqueda por un factor de 100 a 1000 encontrando el vecino más cercano (correcto) el 95% de las veces.
- **Hashing**

$$g: R^d \rightarrow U$$

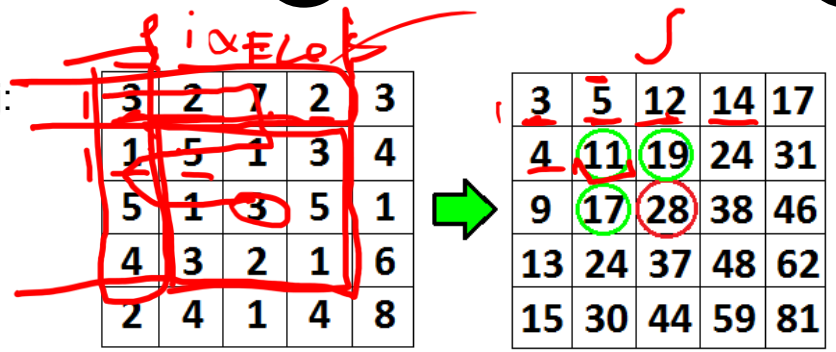
  - **Locality sensitive hashing:** Utiliza uniones de funciones de hashing calculadas independientemente para indexar las características. Además es más sensible a la distribución de parámetros en el espacio.

$$Si\ d(p,q) \leq r \rightarrow P\big(g(p) = g(q)\big) \ldots bastante\ alta$$
$$Si\ d(p,q) > c.r \rightarrow P\big(g(p) = g(q)\big)\ \text{baja}$$

  - También acelera el proceso de búsqueda en dos o tres órdenes de magnitud
- **Transformación Afín**
  - Si logramos detectar tres puntos que sean buenas coincidencias entre una imagen y otra entonces podemos calcular la transformación afín y predecir donde deberían estar los demás descriptores.

Imagen integral

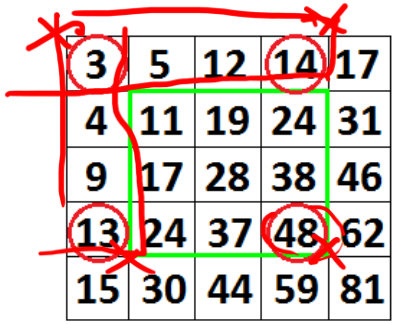
Ej:



3	2	1	2	3
1	5	1	3	4
5	1	3	5	1
4	3	2	1	6
2	4	1	4	8

3	5	12	14	17
4	11	19	24	31
9	17	28	38	46
13	24	37	48	62
15	30	44	59	81

$$s(3,3) = s(2,3) + s(3,2) - s(2,2) + f(3,3)$$
$$s(3,3) = 17 + 19 - 11 + 3 = \mathbf{28}$$



3	5	12	14	17
4	11	19	24	31
9	17	28	38	46
13	24	37	48	62
15	30	44	59	81

$$Suma = 48 + 3 - 14 - 13 = \mathbf{24}$$

$$\sigma = \sqrt{\frac{1}{n}\left(S_2 - \frac{S_1^2}{n}\right)}$$

S1: Suma región rectangular
S2: Suma del cuadrado de esa región
n: Nº de píxeles en la región

- Las imágenes integrales nos permiten calcular de manera eficiente magnitudes como la media, desviación estándar, etc.
- Fue introducido en 1984 por Frank Crow
- Se utilizan en varios ámbitos:
  1. Filtros
  2. Correspondencia de patrones (pattern matching)
  3. Detección de objetos y rostros. Boxlets
  4. Suma de suma de diferencias cuadradas (SSD). Algoritmo estéreo y Motion tracking
- Se obtiene sumando todos los píxeles anteriores, desde la esquina superior izquierda

$$s(i,j) = \sum_{k=0}^{i}\sum_{l=0}^{j} f(k,l)$$

Se puede computar fácilmente utilizando un algoritmo recursivo:

$$\boldsymbol{s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)}$$

- Luego, la suma de píxeles en cualquier región rectangular se puede obtener con una simple operación y de **T constante**.

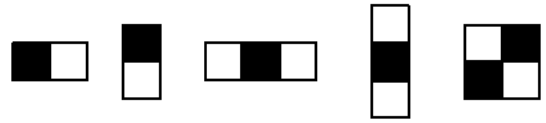
$$\boldsymbol{Suma = Inf_{derecha} + Sup_{izquierda} - Sup_{derecha} - Inf_{izquierda}}$$

- Esto se puede pensar como suma y resta de áreas
- cv2.integral(src[, sdepth]) / cv2.integral2(src[, sdepth[, sqdepth]])

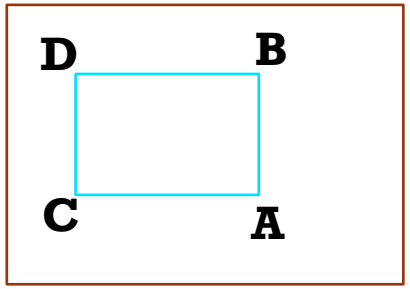
FILTROS DE HAAR

- Buscan detectar patrones de brillo como características de la imagen (diferencias de intensidad entre zonas adyacentes)
- Se pueden definir a múltiples escalas y posiciones
- El resultado del filtro es la diferencia en la suma de los valores de los píxeles entre zonas claras y oscuras
- Se pueden computar muy rápidamente utilizando imágenes integrales, solo tres operaciones de suma para encontrar el área de cualquier tamaño de rectángulo
- Considerando todos los posibles parámetros (posición y escala y tipo) de estos filtros se pueden armar en imágenes de 24x24 más de 180.000 características

Filtros de Haar Básicos

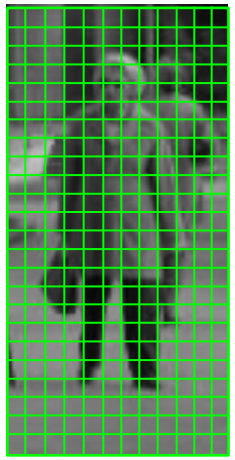
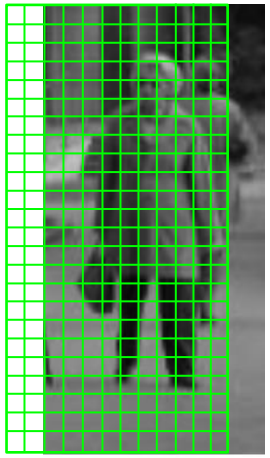


200	200	100	100	200	200	100	100
250	250	50	50	250	250	50	50
255	255	255	255	100	100	100	100
255	255	255	255	100	100	100	100
200	200	100	100	200	200	100	100
250	250	50	50	250	250	50	50
255	255	255	255	100	100	200	200
255	255	255	255	100	100	250	250

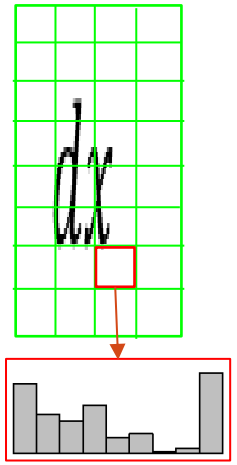


Suma= A-B-C+D

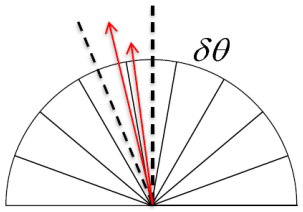
HOG (HISTOGRAM OF GRADIENTS)

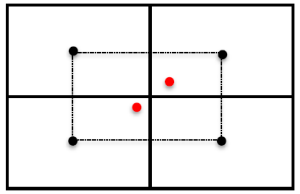






dx dy M_g





- Para imágenes color se calculan los gradientes en cada canal (R, G, B) y se selecciona el gradiente del canal con mayor magnitud
- Las orientaciones se agrupan en intervalos (bins) con y sin signo
- Histograma:

$$h(k) = \sum_{(x,y)\in C} w_k(x,y) * M_g(x,y)$$

$$w(k) = \begin{cases} 1, & si\ (k-1)\delta\theta \le \theta(x,y) < k\delta\theta \\ 0, & en\ caso\ contrario \end{cases}$$

Para robustecer el descriptor:

1. Pixeles con orientaciones similares se asignan con pesos proporcionales a su distancia a la frontera del intervalo.
2. También se pondera según la ubicación (x,y) dentro del bloque.
3. Se aplica una normalización en bloques para robustecer frente a cambios de intensidad

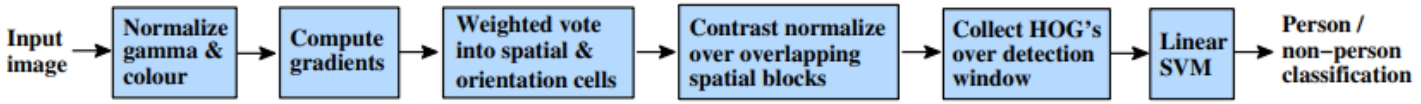
Descriptor final

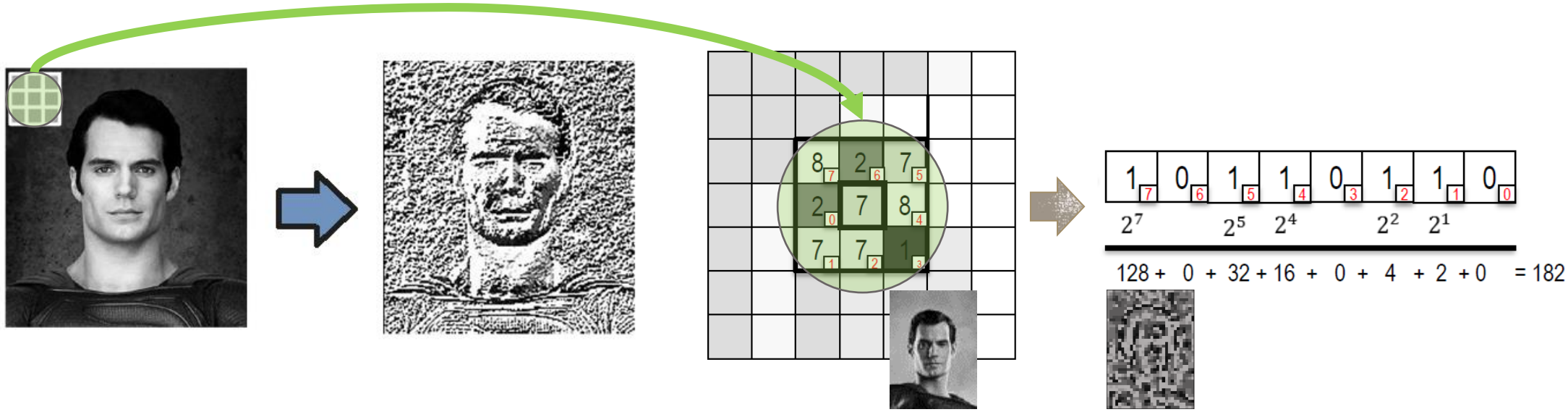
- Tamaño de celda ($n\ x\ m$)
- N° de intervalos de las orientaciones
- N° de celdas en cada bloque con superposición.

$$HOG = (X_1, \dots, X_n)\ ,\ n = n°bloques\ x\ n°celdas\ x\ n\ bins$$

Valores óptimos

- Resolución de imagen: 64 x 128 (N. Dalal and B. Triggs, 2005)
- Celdas de 8x8 para calculo de histograma
- Normalizar en bloques de 16x16



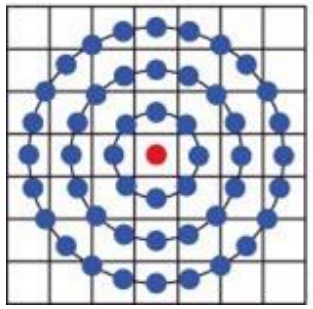


LBP (TEXTURA)

- LBP: Local Binary Pattern

1. Invariante frente a cambios monotónicos de nivel de gris (brillo)
2. Invariante a desplazamientos

$$LBP = \sum_{p=0}^{p-1} s(g_p - g_c)2^p \quad ; \quad s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$



U = 0

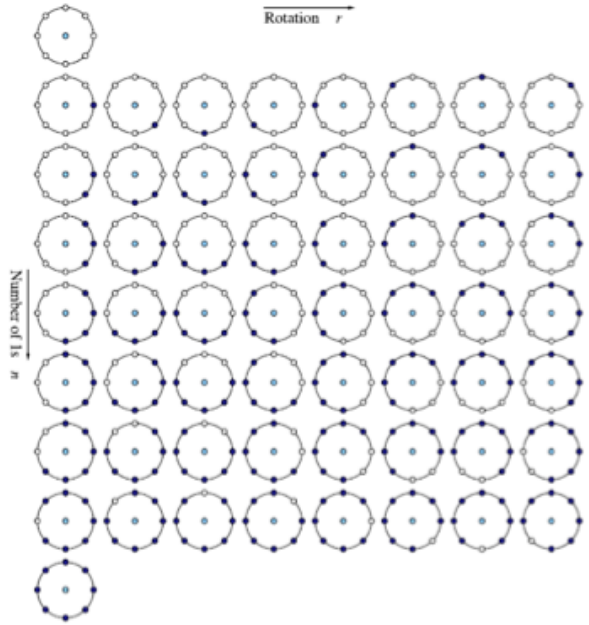
1	1	1
1	·	1
1	1	1

U = 2

0	0	0
1	·	1
1	1	1

U = 4

1	1	0
1	·	1
0	1	1

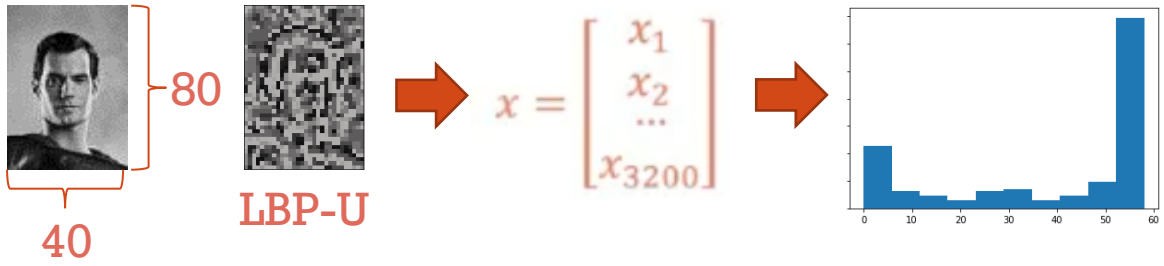


LBP - VARIANTES

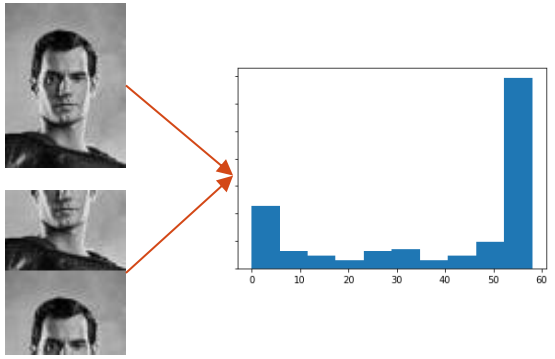
- Definición de vecindad
  - Determinada por el radio: R=1, R=2, R=3, etc…
- Comparación con un umbral (t)

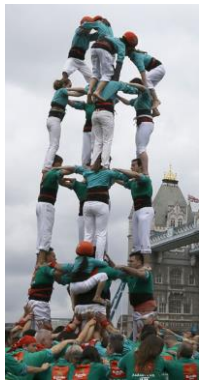
$$LBP = \sum_{p=0}^{p-1} s(g_p - g_c)2^p \quad ; \; s(x) = \begin{cases} 0, & x < t \\ 1, & x \geq t \end{cases}$$

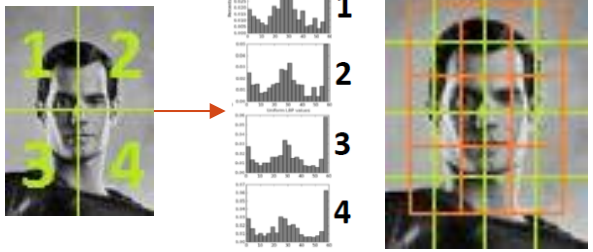
- LBP-U
  - Objetivo: Reducir numero de patrones.
  - U: Medida de uniformidad (N° de transiciones $0 \leftrightarrow 1$)
    - A los patrones con U=0 o U = 2 les asignamos un código de patrón individual
    - Al resto de los patrones se les asigna el mismo código (pasan a ser indistinguibles)
    - LBP=256 patrones, LBP-U: (58 + 1) patrones



Problema…







LBP

- Si nos quedamos sólo con los LBP por cada píxel
  1. tenemos NxM características
  2. Comparando punto a punto el alineamiento de la ventana sería crítico!
- Entonces?
  - Hacemos un histograma…*y pasamos de 3200 a 256 características!*
  - Y si usamos LBP-U…*pasamos de 3200 a 59 características!!*
- Problemas con las ventanas
  - Detecciones múltiples
  - Solución:
    - Dividir en subventanas: LBP-U por bloques
    - Cada histograma debe coincidir en forma y posición.
    - En la práctica además de la subdivisión hay solape → caso Superman:

$Nº\ carac.\ ej: (4 \times 4 + 3 \times 3) * 59 = 1475$

$Nº\ carac.\ a\ impl.: (4 + 1) * 59 = \mathbf{295}$

TP4

- Para las imágenes suministradas:
  1. Implementar un extractor de características LBP básico (sin uniformidad, 8 vecinos)
  2. Realizar el histograma de características LBP de la imagen
  3. Comparar los histogramas