Detectores y Descriptores

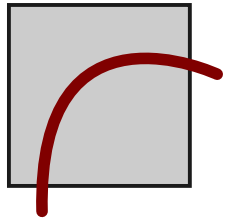
Araguás, Gastón Redolfi, Javier

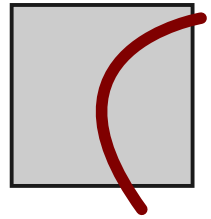
3 de junio del 2020

Detectores invariantes a escala

La clase anterior vimos detectores.. (Harris, Shi-Tomasi) que son:

- Invariantes a rotación



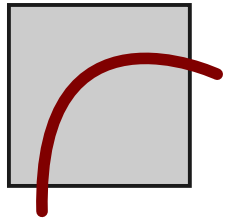


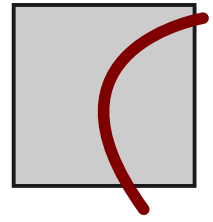
$$H = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad H' = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R^{-1}$$

Detectores invariantes a escala

La clase anterior vimos detectores.. (Harris, Shi-Tomasi) que son:

- Invariantes a rotación





$$H = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad H' = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R^{-1}$$

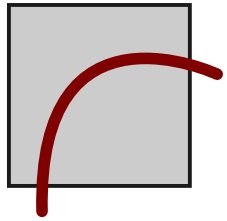
- Invariantes (robustos) a cambios de iluminación

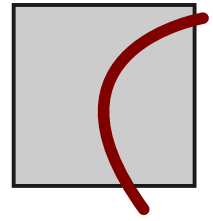
$$I \to I + b \qquad \text{No afecta a derivadas...}$$

Detectores invariantes a escala

La clase anterior vimos detectores.. (Harris, Shi-Tomasi) que son:

- Invariantes a rotación



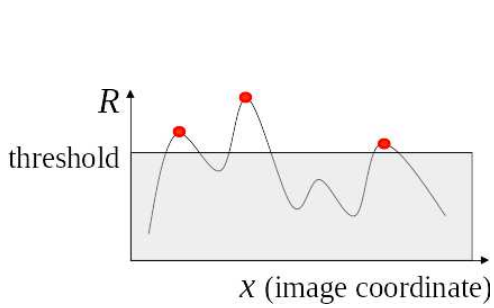


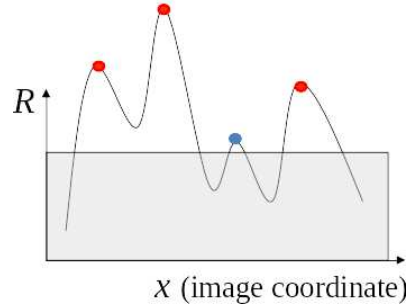
$$H = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad H' = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R^{-1}$$

- Invariantes (robustos) a cambios de iluminación

$I \to I + b$ No afecta a derivadas...

$I \to aI$

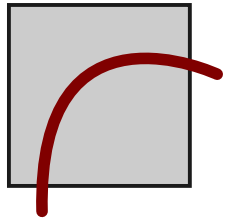


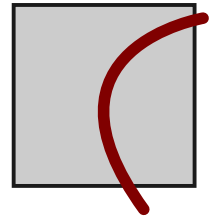


Detectores invariantes a escala

La clase anterior vimos detectores.. (Harris, Shi-Tomasi) que son:

- Invariantes a rotación



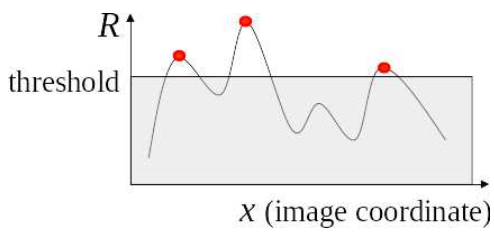


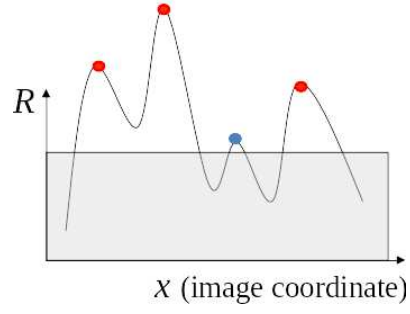
$$H = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad H' = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R^{-1}$$

- Invariantes (robustos) a cambios de iluminación

$I \to I + b$ No afecta a derivadas...

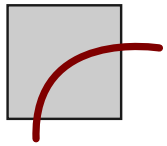
$I \to aI$

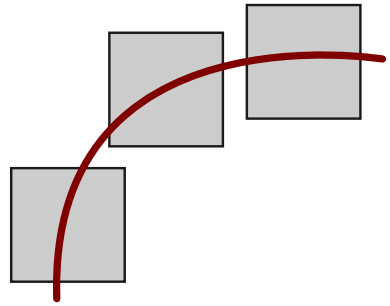




- pero ...son invariantes a escala?

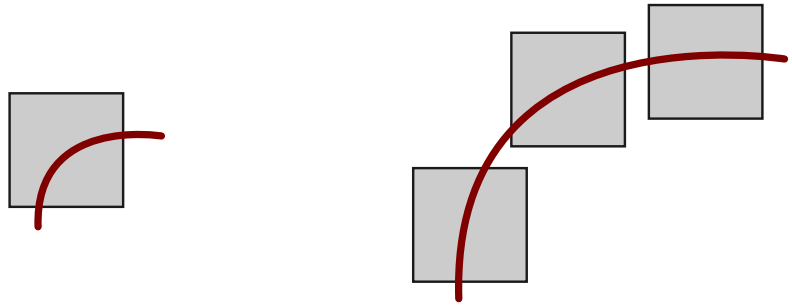
Detectores invariantes a escala





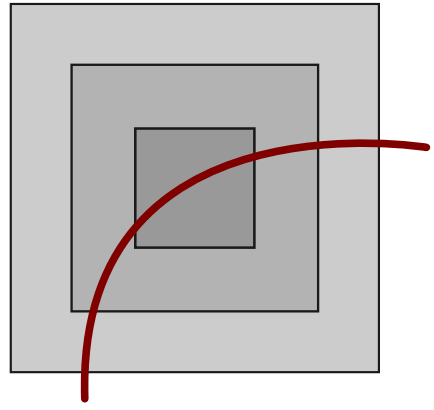
Nop…

Detectores invariantes a escala



Nop...

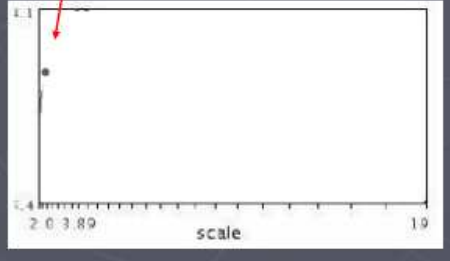
Como solucionarlo?



- Buscar la esquina en muchas escalas (tamaños de ventana)
- Definir una función f de "esquinidad" en función de la escala.
- Encontrar la escala que maximice la función f.
- La función f puede ser el operar Harris ($R = det(H) - k.traza(H)$).

Detectores invariantes a escala

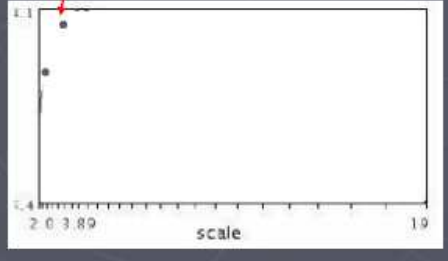
Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Detectores invariantes a escala

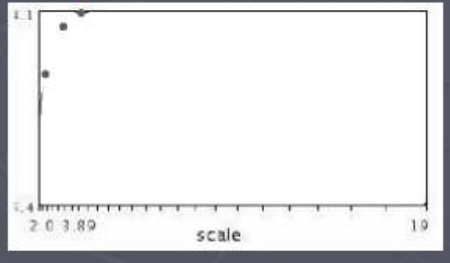
Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Detectores invariantes a escala

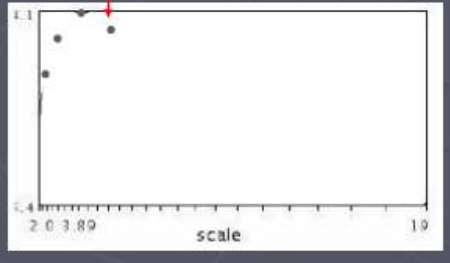
Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

Detectores invariantes a escala

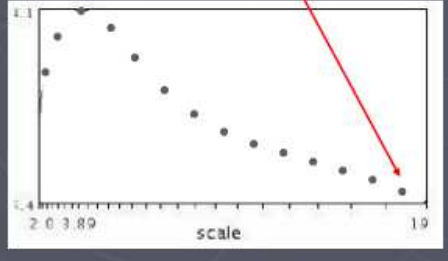
Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Detectores invariantes a escala

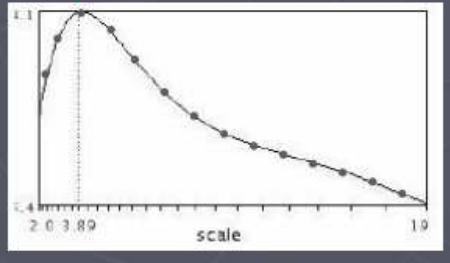
Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

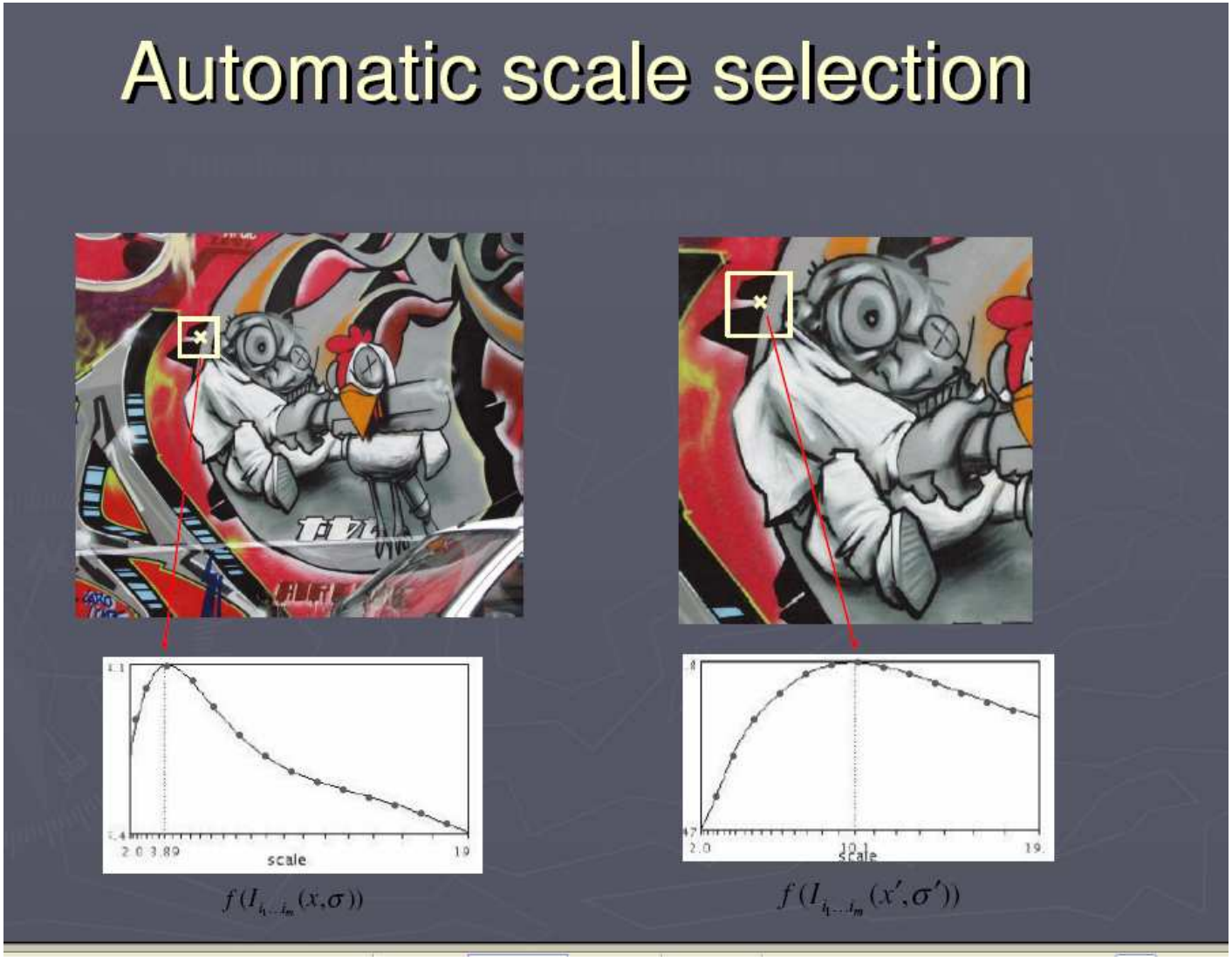
Detectores invariantes a escala

Automatic scale selection

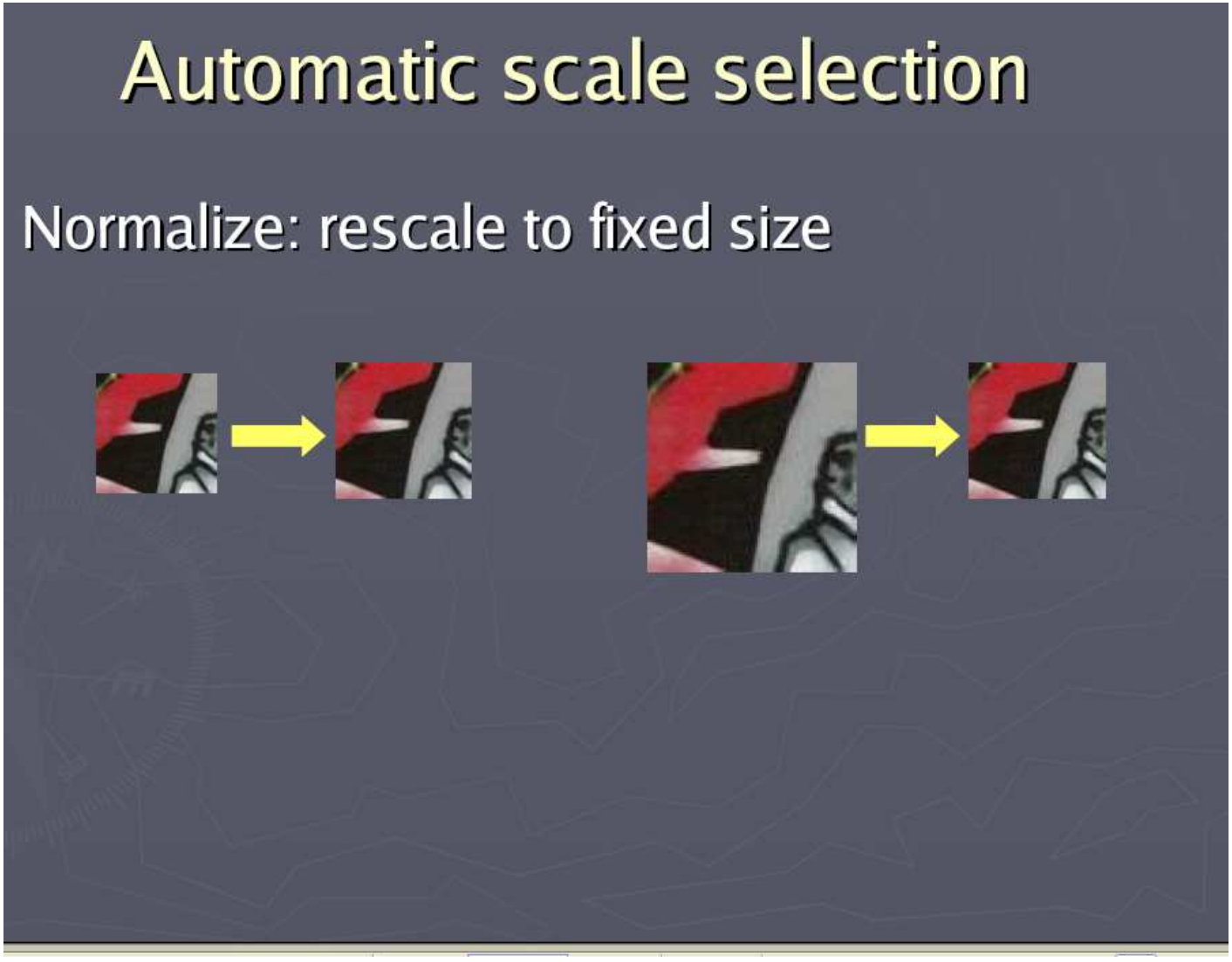


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Detectores invariantes a escala

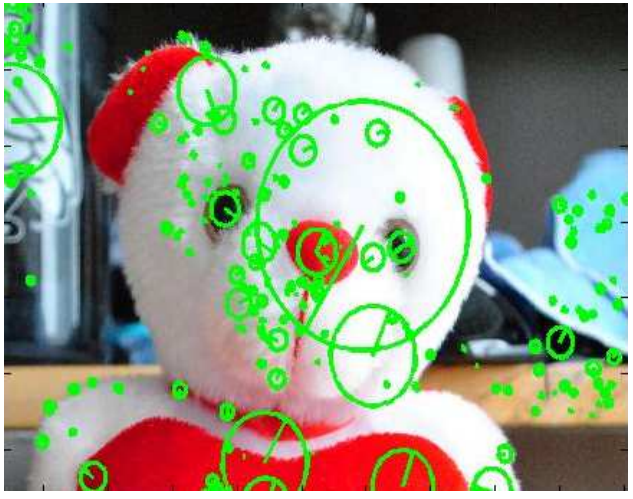


Detectores invariantes a escala



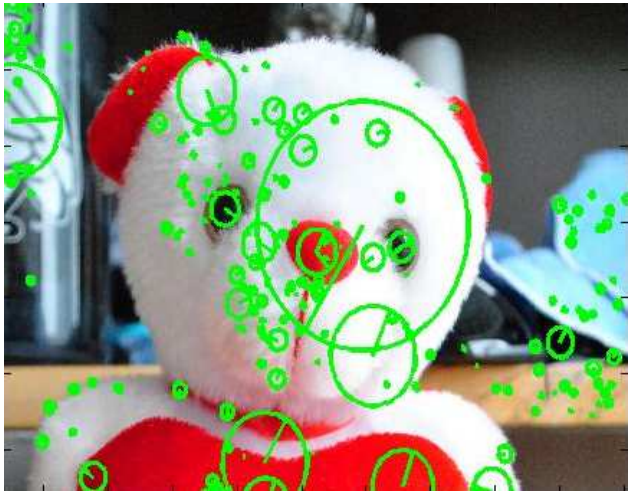
Descriptor

Un detector invariante a escala determina la posición y escala de la característica saliente en la imagen $(u, v, scale)$.

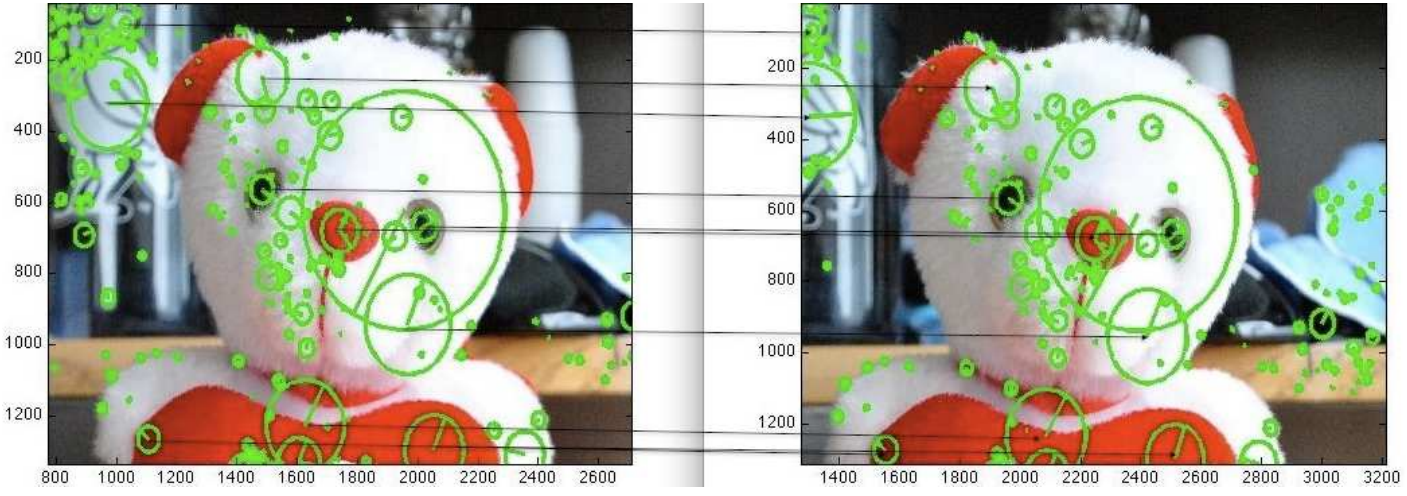


Descriptor

Un detector invariante a escala determina la posición y escala de la característica saliente en la imagen $(u, v, scale)$.

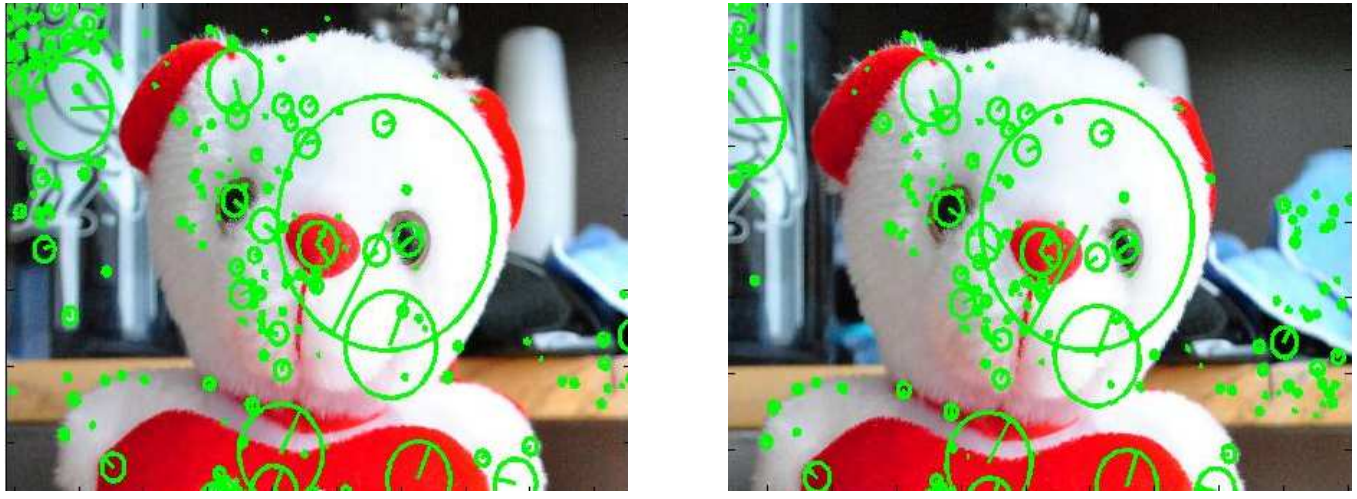


Pero, ¿Cómo buscar correspondencias?



Descriptor

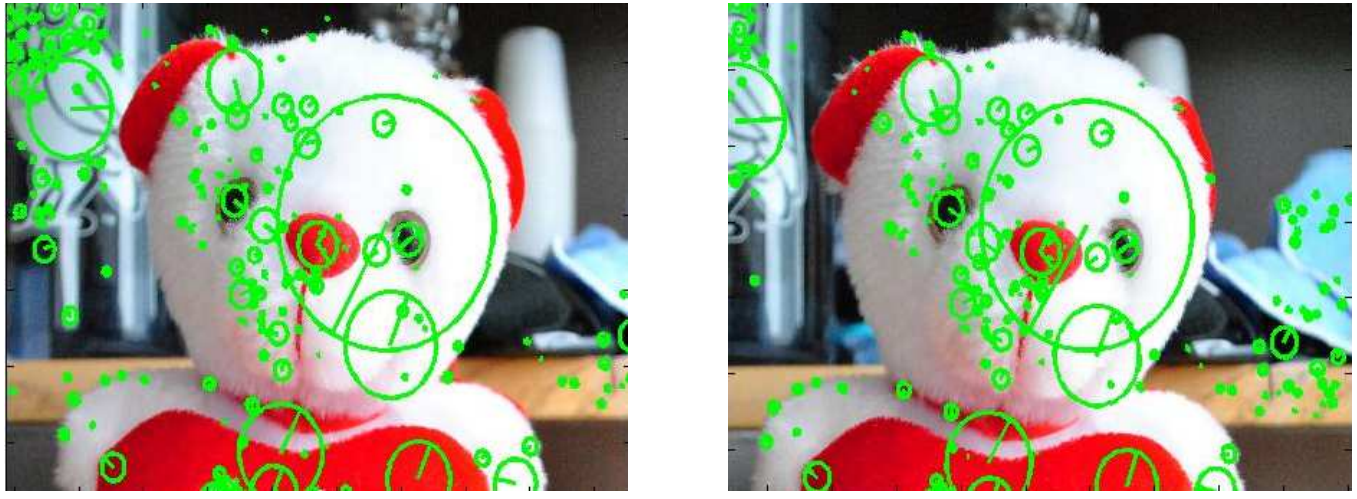
Un detector invariante a escala determina la posición y escala de la característica saliente en la imagen $(u, v, scale)$.



Pero, ¿Cómo buscar correspondencias?

Descriptor

Un detector invariante a escala determina la posición y escala de la característica saliente en la imagen $(u, v, scale)$.

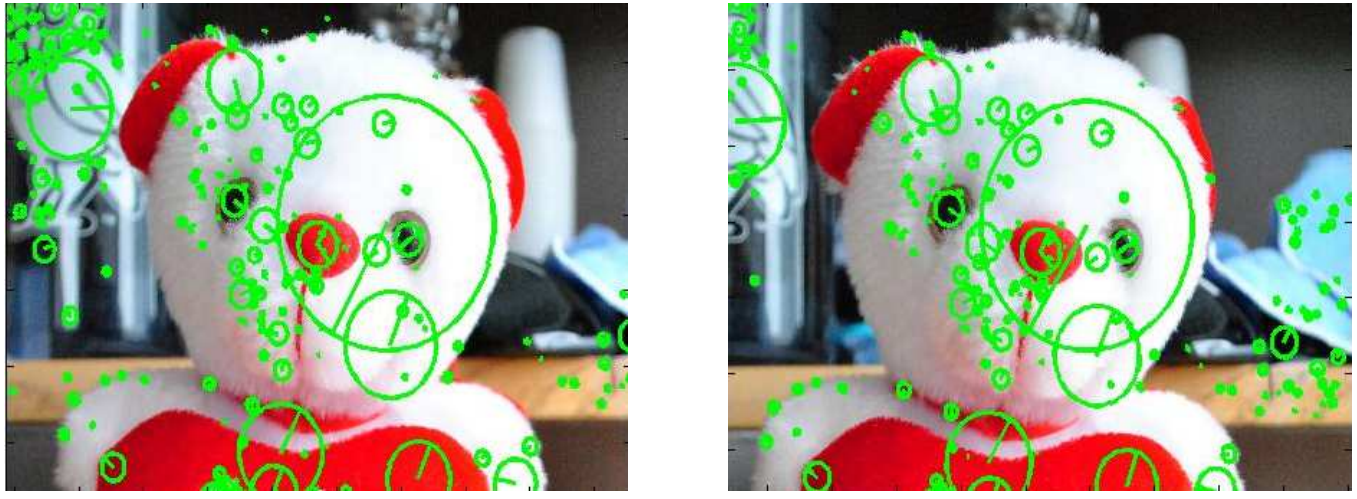


Pero, ¿Cómo buscar correspondencias?

- Por posición? -> sólo para pequeños movimientos.

Descriptor

Un detector invariante a escala determina la posición y escala de la característica saliente en la imagen $(u, v, scale)$.

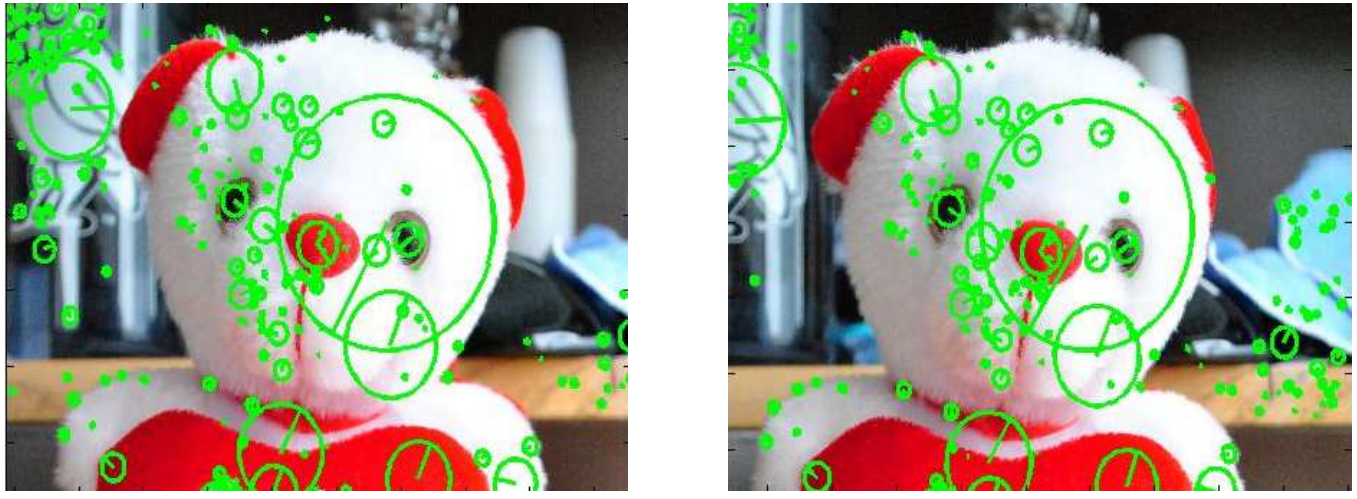


Pero, ¿Cómo buscar correspondencias?

- Por posición? -> sólo para pequeños movimientos.
- Por parches? -> muy sensible a cambios de iluminación, vistas, etc.

Descriptor

Un detector invariante a escala determina la posición y escala de la característica saliente en la imagen $(u, v, scale)$.

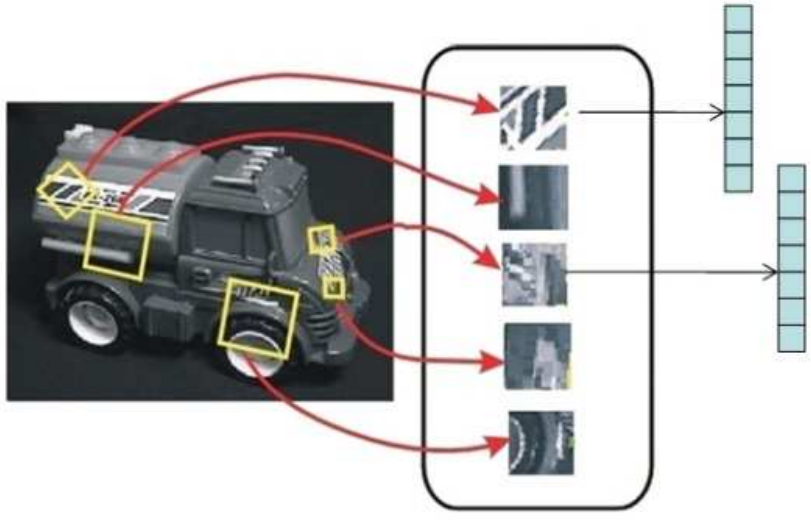


Pero, ¿Cómo buscar correspondencias?

- Por posición? -> sólo para pequeños movimientos.
- Por parches? -> muy sensible a cambios de iluminación, vistas, etc.
- . . . lo que necesitamos es un descriptor.

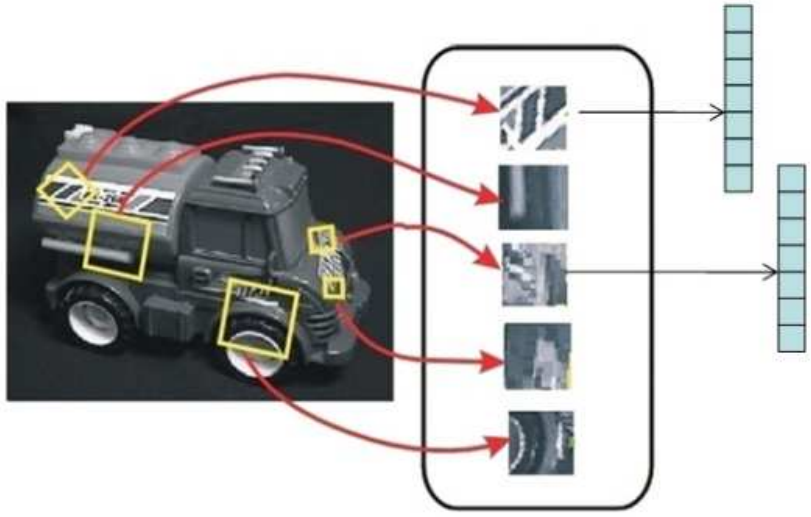
Descriptor

Un descriptor es un representación numérica de la zona alrededor de una característica de imagen,

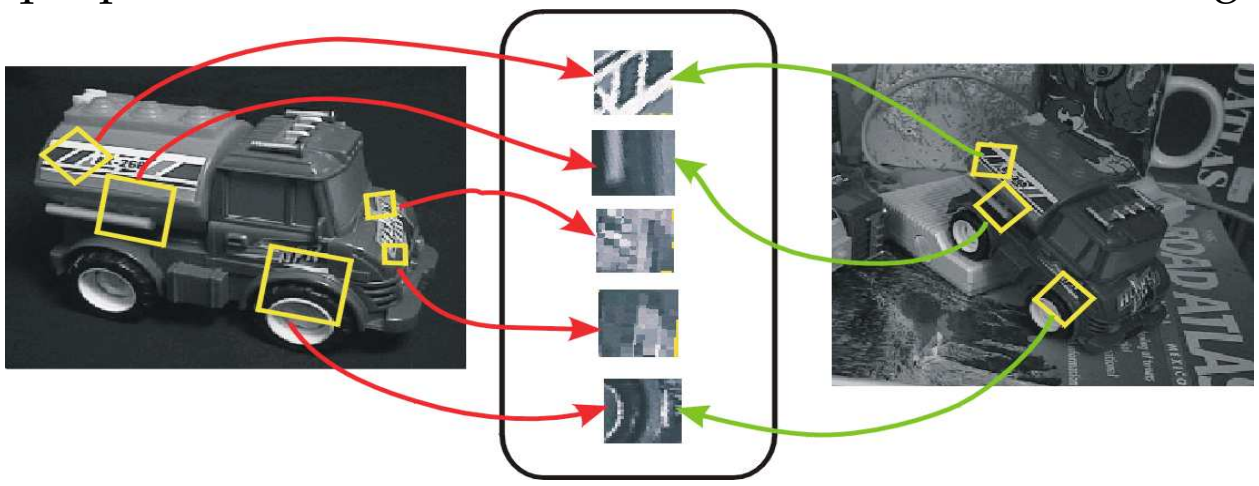


Descriptor

Un descriptor es un representación numérica de la zona alrededor de una característica de imagen,



que permite identificar la característica en distintas imágenes.



Descriptor

Un descriptor es un representación numérica de la zona alrededor de una característica de imagen,

Incluso en escenarios muy complejos...

Tipos de descriptores

- Basados en intensidad
- Basados en histogramas
- Basados en gradientes
- Basados en color
- Basados en frecuencia
- Basados en forma
- Combinación de varios de los anteriores

Descriptores más comunes

- SIFT
- SURF
- FAST
- BRIEF
- ORB

Descriptores más comunes

SIFT - Scale Invariant Feature Transform

- SIFT
- SURF
- FAST
- BRIEF
- ORB

D. Lowe. Distinctive image features from scale-invariant key points. International Journal of Computer Vision 2004.

Descriptores más comunes

- SIFT
- SURF
- FAST
- BRIEF
- ORB

SIFT - Scale Invariant Feature Transform

- Basado en histograma de gradientes

D. Lowe. Distinctive image features from scale-invariant key points. International Journal of Computer Vision 2004.

Descriptores más comunes

- SIFT
- SURF
- FAST
- BRIEF
- ORB

SIFT - Scale Invariant Feature Transform

- Basado en histograma de gradientes
- Es además un detector que agrega orientación (x, y, s, θ)

D. Lowe. Distinctive image features from scale-invariant key points. International Journal of Computer Vision 2004.

Descriptores más comunes

- SIFT
- SURF
- FAST
- BRIEF
- ORB

SIFT - Scale Invariant Feature Transform

- Basado en histograma de gradientes
- Es además un detector que agrega orientación (x, y, s, θ)
- Algorítmo patentado por la Universidad de British Columbia

D. Lowe. Distinctive image features from scale-invariant key points. International Journal of Computer Vision 2004.

Descriptores más comunes

- SIFT
- SURF
- FAST
- BRIEF
- ORB

SIFT - Scale Invariant Feature Transform

- Basado en histograma de gradientes
- Es además un detector que agrega orientación (x, y, s, θ)
- Algorítmo patentado por la Universidad de British Columbia
- Expira este año 2020, ya hay una idea para el Google Summer code 2020

IDEA: Better SIFT in the main repository

Description: In 2020 March the patent on one of the most popular feature detection algorithm, SIFT, expires. So, we can move the implementation from `opencv_contrib/xfeatures2d` to the main OpenCV repository (`opencv/features2d`) in the late spring or summer. We can also optimize and improve it further, probably create bit-exact implementation.

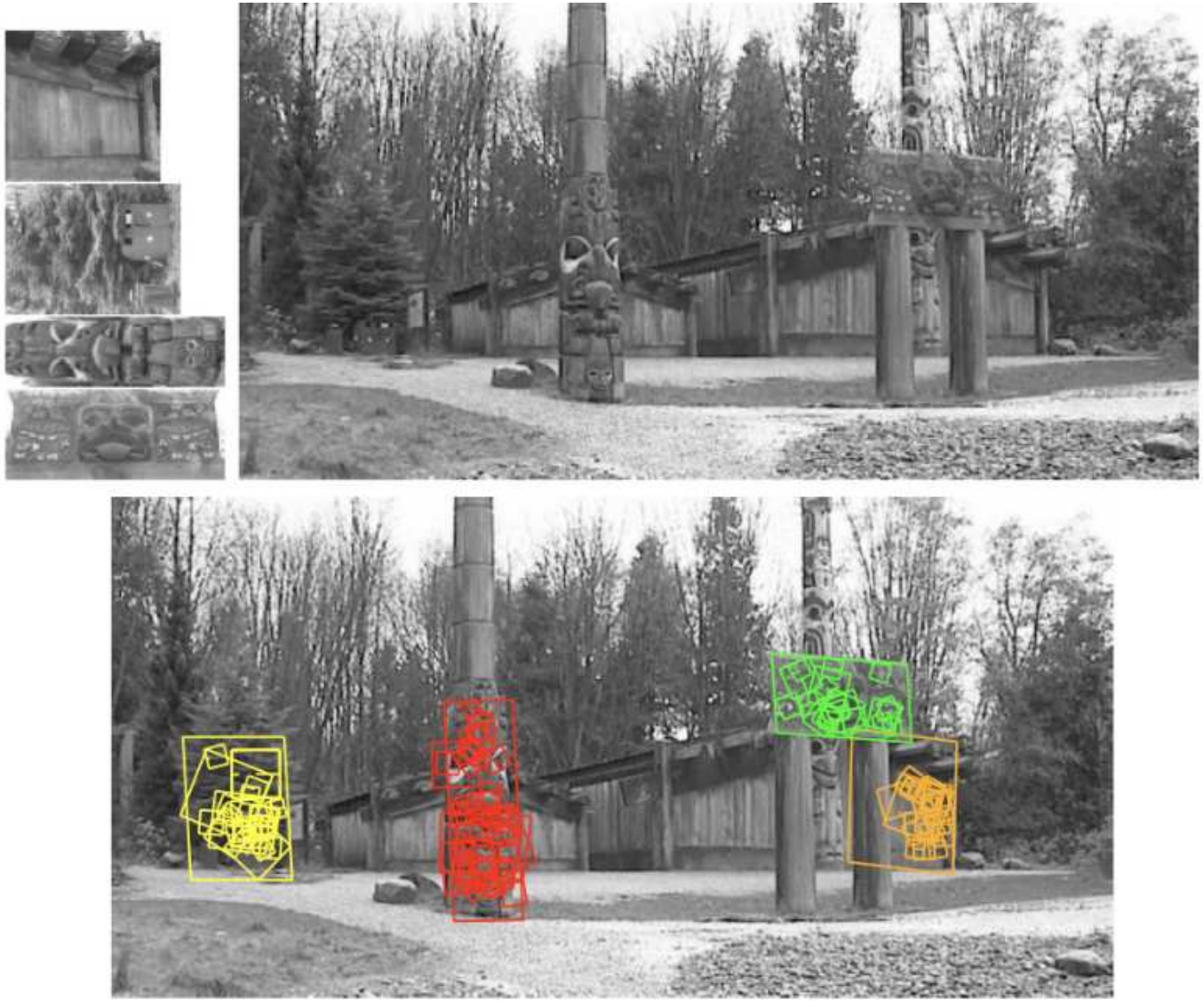
D. Lowe. Distinctive image features from scale-invariant key points. International Journal of Computer Vision 2004.

SIFT - Scale Invariant Feature Transform

Porque tanto lío con SIFT?

SIFT - Scale Invariant Feature Transform

Porque tanto lío con SIFT?



SIFT - Scale Invariant Feature Transform

- Detección de salientes en espacio de escala
- Localización de características
- Asignación de dirección de la características
- Desrotación de gradientes de cada característica
- Cálculo de histograma de gradientes de subregiones de características
- Medición de distancia entre características

SIFT - Scale Invariant Feature Transform

- Detección de salientes en espacio de escala
- Localización de características
- Asignación de dirección de la características

Detector

- Desrotación de gradientes de cada característica
- Cálculo de histograma de gradientes de subregiones de características

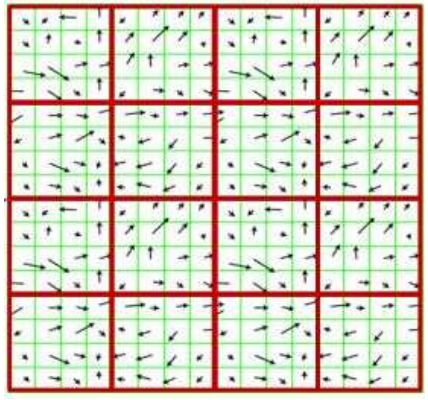
Descriptor

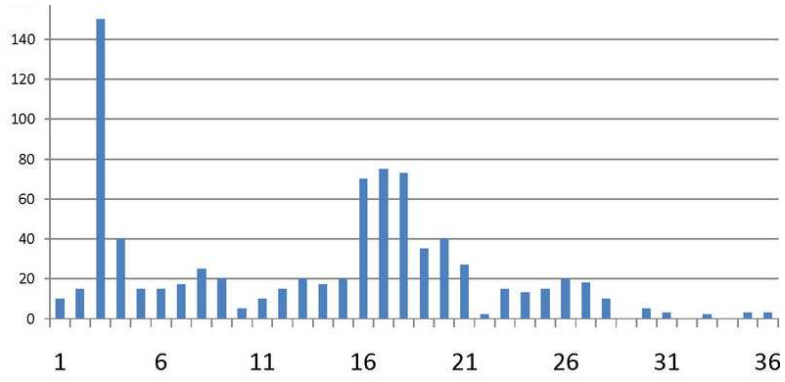
- Medición de distancia entre características

Matching

SIFT - Descriptor invariante a rotación

Por cada característica detectada (similar a Harris pero multiescala)



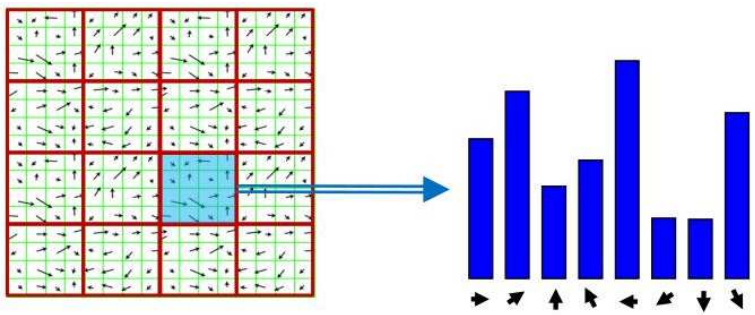


- Se realiza un histograma de orientaciones con los pixeles que rodean al punto de interés.
- El histograma contiene 36 bins que codifican los 360°.
- El pico más alto del histograma será considerado la dirección de la característica (x, y, s, θ).
- Si hay otros picos también se consideran (mayores al 80 % del máximo), generando una nueva característica (x, y, s, θ_2).

Descriptor SIFT

Cómputo del descriptor

- Se calcula magnitud y orientación del gradiente.
- Los gradientes se orientan teniendo en cuenta la orientación del punto de interés.
- La región del punto de interés se divide en una grilla de por ejemplo 4x4.
- En cada celda de la grilla se calcula un histograma de orientaciones con 8 bins.

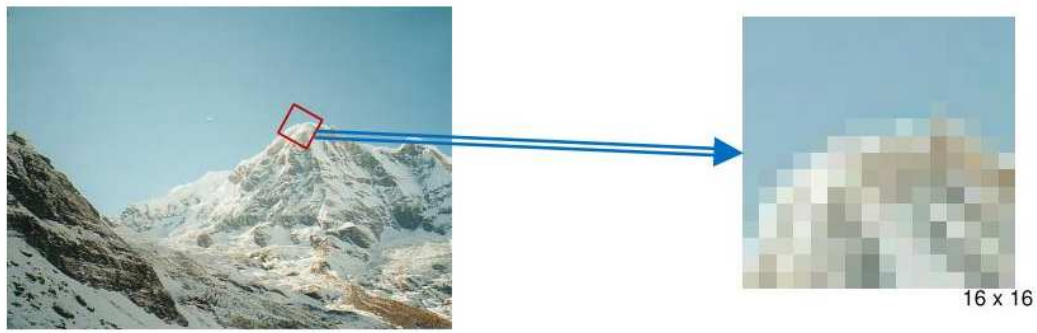


- Nos da un vector de 128 dimensiones, 4x4 regiones por 8 bins.

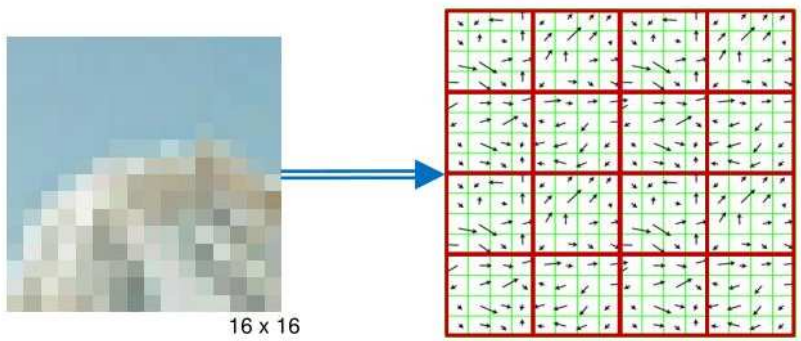
Descriptor SIFT

Cómputo de SIFT en una imagen:

- Se detecta la característica, se la reescala a 16x16 y se elimina la rotación



- Se divide en regiones de 4x4, se calculan los histogramas de gradientes y se construye el descriptor



- El vector puede ser comparado con otros simplemente midiendo distancia euclidia (norma L_2).

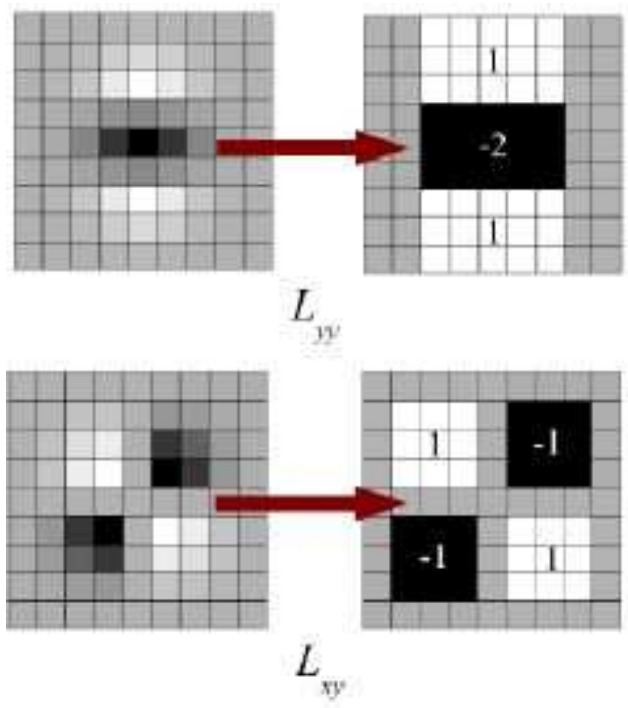
Ejemplo de uso de SIFT - Código

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import cv2

img = cv2.imread('img2.ppm')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

sift = cv2.xfeatures2d.SIFT_create()
kp, dscr = sift.detectAndCompute(gray, None)

cv2.drawKeypoints(img, kp, img)

cv2.imwrite('sift_keypoints.png', img)

cv2.imshow('sift_keypoints', img)
cv2.waitKey()
```

SURF - Speeded Up Robust Features

(No vamos a entrar en detalles...)

- Versión rápida de SIFT (3 veces más rápido)
- Se aproxima el LoG con un Box Filter (filtro de caja)
  - Puede ser calculado con imágenes integrales
    `https://theailearner.com/tag/integral-image-opencv/`
  - Es fácilmente paralelizable (escalas)



Ejemplo de uso de SURF - Código

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import cv2

img = cv2.imread('img2.ppm')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

surf = cv2.xfeatures2d.SURF_create()
kp, dscr = surf.detectAndCompute(gray, None)

cv2.drawKeypoints(img, kp, img)

cv2.imwrite('surf_keypoints.png', img)

cv2.imshow('surf_keypoints', img)
cv2.waitKey()
```

Descriptores y detectores en OpenCV

En OpenCV también están disponibles

- SURF-extended (vector de 128D)
- U-SURF (no es invariante a rotación)
- FAST (Features from Accelerated Segment Test)
- BRIEF (Binary Robust Independent Elementary Features)
- ORB (Oriented FAST and Rotated BRIEF)

Todos se usan de forma similar a SIFT, y tienen distintas ventajas y desventajas, dependiendo de la aplicación.

Práctico 11 - Alineación usando SIFT

Alineación de imágenes usando SIFT

1. Capturar dos imágenes con diferentes vistas del mismo objeto
2. Computar puntos de interés y descriptores en ambas imágenes
3. Establecer matches entre ambos conjuntos de descriptores
4. Eliminar matches usando criterio de Lowe [1]
5. Computar una homografía entre un conjunto de puntos y el otro
6. Aplicar la homografía sobre una de las imágenes y guardarla en otra (mezclarla con un alpha de 50 %)

[1] Se calcula el cociente entre las distancias a los 2 vecinos más cercanos de cada descriptor y si es mayor que 0.8 se descarta el match

Práctico 11 - Puntos claves en ambas imágenes

Práctico 11 - Matches entre ambas imágenes

Práctico 11 - Mezcla de ambas imágenes

Práctico 11 - Código de Ayuda

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import cv2

MIN_MATCH_COUNT = 10

img1 = # Leemos la imagen 1
img2 = # Leemos la imagen 2

dscr = # Inicializamos el detector y el descriptor

kp1, des1 = # Encontramos los puntos clave y los descriptores con SIFT en la imagen 1
kp2, des2 = # Encontramos los puntos clave y los descriptores con SIFT en la imagen 2

matcher = cv2.BFMatcher(cv2.NORM_L2)

matches = matcher.knnMatch(des1, des2, k=2)

# Guardamos los buenos matches usando el test de razón de Lowe
good = []
for m, n in matches:
    if m.distance < 0.7*n.distance:
        good.append(m)

if(len(good) > MIN_MATCH_COUNT):
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
    # Computamos la homografía con RANSAC
    H, mask = cv2.findHomography(dst_pts, src_pts, cv2.RANSAC, 5.0)

wimg2 = # Aplicamos la transformación perspectiva H a la imagen 2

# Mezclamos ambas imágenes
alpha = 0.5
blend = np.array(wimg2 * alpha + img1 * (1 - alpha), dtype=np.uint8)
```