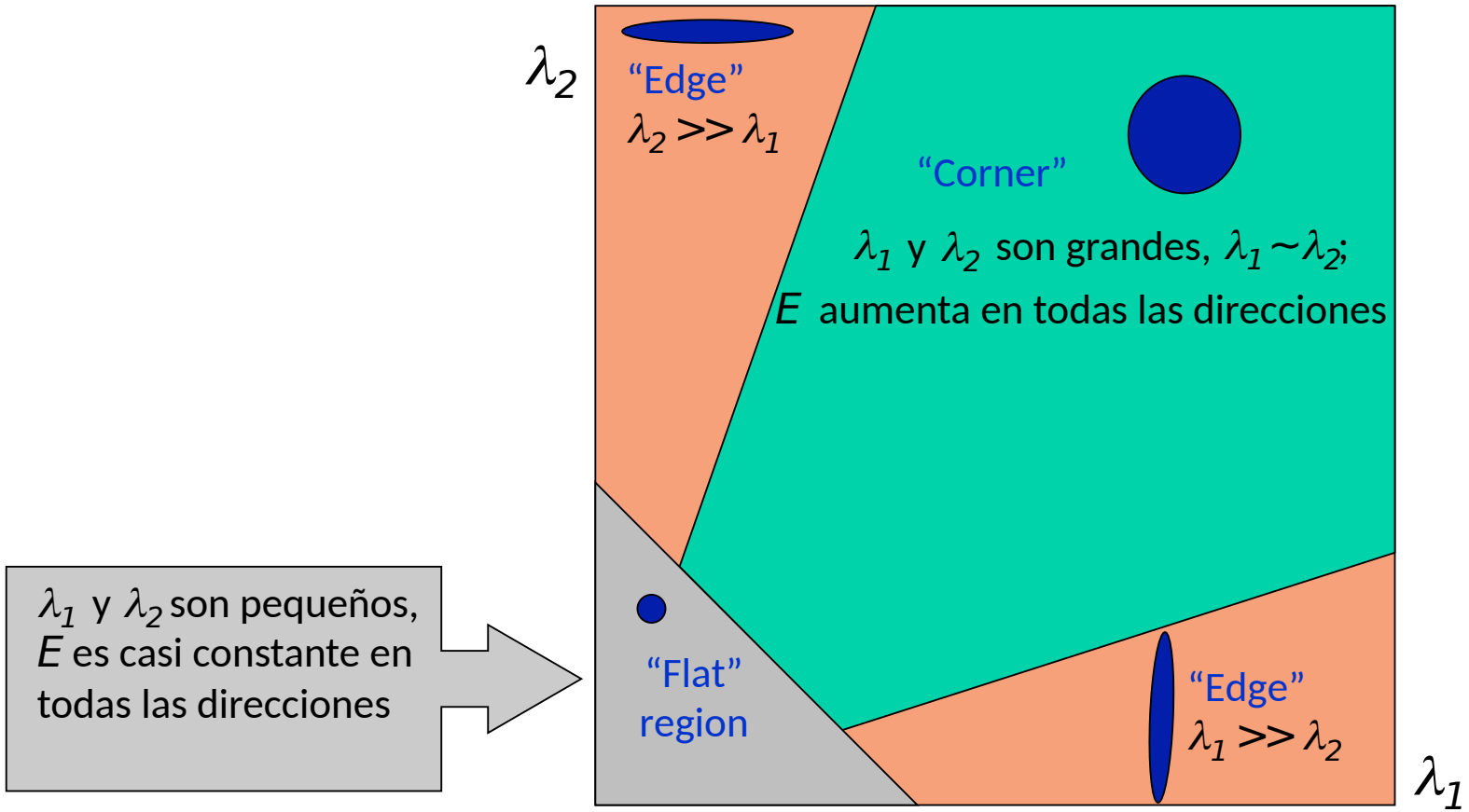
Detectores

Araguás, Gastón Redolfi, Javier

22 de Mayo del 2019

Detectores de esquinas de Harris



Detectores de esquinas de Harris

Se basa en la ecuación:

$$R = det(M) - k\ \ trace(M)^2 = \lambda_1\lambda_2 - k\ \ (\lambda_1 + \lambda_2) \tag{1}$$

Utilizamos el método

```
dst = cv2.cornerHarris(src, blockSize, ksize, k)
```

- **`src`**: es la imagen de entrada
- **`blockSize`**: tamaño de la ventana
- **`ksize`**: tamaño del kernel de Sobel
- **`k`**: parámetro libre de la ecuación

Ejemplo de detector de esquinas de Harris

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


import cv2
import numpy as np


filename = 'pattern.png'
img = cv2.imread(filename)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = np.float32(gray)

dst = cv2.cornerHarris(gray, 2, 3, 0.04)

dst = cv2.dilate(dst, kernel=None)

img[dst > 0.01*dst.max()] = [0, 0, 255]

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

cv2.imshow('dst', img)

if(cv2.waitKey(0) & 0xff == 27):
    cv2.destroyAllWindows()
```

Esquinas con precisión subpixel

Utilizamos el método

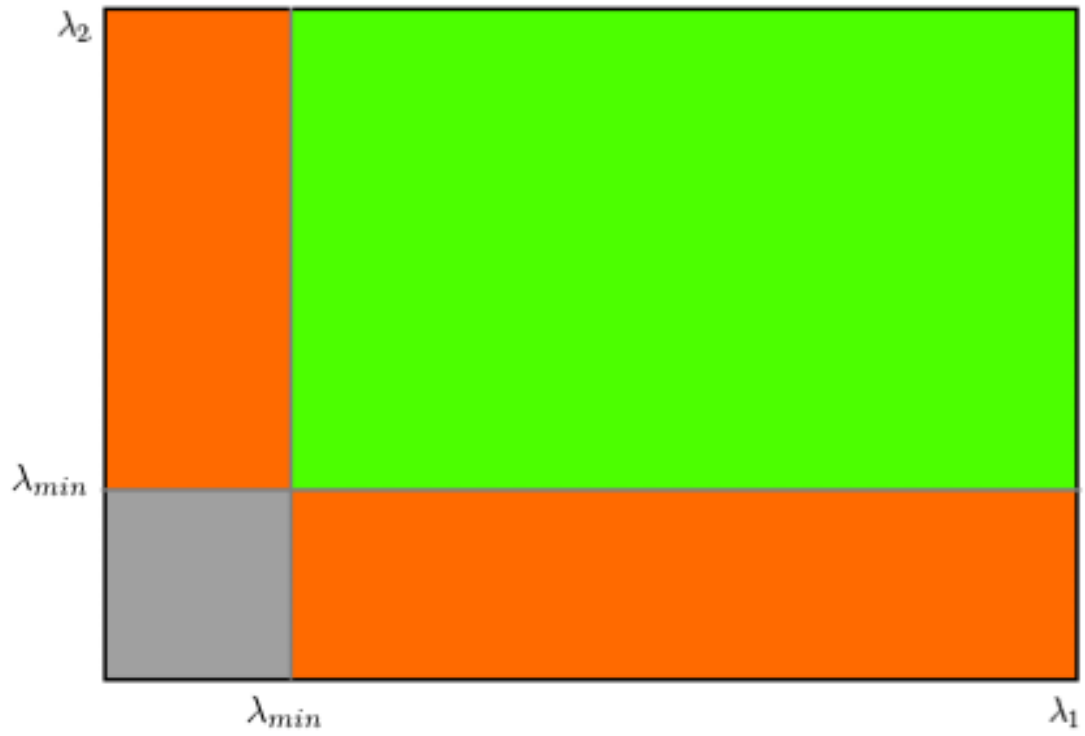
```
corners2 = cv2.cornerSubPix(image, corners, winSize, zeroZone,
criteria)
```

- **`image`** imagen en donde encontramos las esquinas
- **`corners`** ubicaciones iniciales de las esquinas en la imagen
- **`winSize`** tamaño de la ventana de refinamiento
- **`zeroZone`** ?
- **`criteria`** criterio de terminación
- **`corners2`** ubicaciones refinadas de las esquinas en la imagen

Ejemplo de esquinas con precisión subpixel

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

import cv2
import numpy as np

filename = 'graffiti.jpg'
img = cv2.imread(filename)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

gray = np.float32(gray)
dst = cv2.cornerHarris(gray, blockSize=2, ksize=3, k=0.04)
dst = cv2.dilate(dst, kernel=None)
ret, dst = cv2.threshold(dst, 0.01*dst.max(), 255, 0)
dst = np.uint8(dst)

# Encontramos centroides
ret, labels, stats, centroids = cv2.connectedComponentsWithStats(dst)

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.001)
corners = cv2.cornerSubPix(gray, np.float32(centroids), (5, 5), (-1, -1),
                           criteria)

res = np.hstack((centroids, corners))
res = np.int0(res)
img[res[:, 1], res[:, 0]] = [0, 0, 255]
img[res[:, 3], res[:, 2]] = [0, 255, 0]
cv2.imwrite('harris_subpixel.png', img)

cv2.imshow('dst', img)
if(cv2.waitKey(0) & 0xff == 27):
    cv2.destroyAllWindows()
```

Detectores de esquinas de Shi-Tomasi



Detectores de esquinas de Shi-Tomasi

El detector de esquinas de Shi-Tomasi encuentra las esquinas más prominentes en la imagen.
Proponen una modificación al detector de esquinas de Harris:

$$R = min(\lambda_1, \lambda_2) \tag{2}$$

- Se calcula la medida de calidad de esquina en cada pixel de la imagen usando el mínimo eigen-valor.
- Se aplica supresión de no-máximo en una vecindad de 3x3.
- Las esquinas con eigen-valor menor que el nivel de calidad son rechazados.
- Las esquinas que quedaron se ordenan en forma descendiente con respecto a su calidad.
- Por último se descartan las esquinas que tienen una esquina de mayor calidad a una distancia menor a la distancia mínima.

Detectores de esquinas de Shi-Tomasi

Utilizamos el método
`corners = cv2.goodFeaturesToTrack(image, maxCorners, qualityLevel, minDistance)`

- **`image`**: es la imagen de entrada
- **`maxCorners`**: número máximo de esquinas
- **`qualityLevel`**: mínimo nivel de calidad de las esquinas
- **`minDistance`**: distancia mínima posible entre esquinas

Ejemplo de detector de esquinas de Shi-Tomasi

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

import numpy as np
import cv2

img = cv2.imread('opencv-logo2.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

corners = cv2.goodFeaturesToTrack(gray, maxCorners=50,
                                  qualityLevel=0.01,
                                  minDistance=10)
corners = np.int0(corners)

for i in corners:
    x, y = i.ravel()
    cv2.circle(img, (x, y), 3, 255, -1)

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

cv2.imshow('dst', img)
if(cv2.waitKey(0) & 0xff == 27):
    cv2.destroyAllWindows()
```