Shi-Tomashi

filip.reichl, tomas.adam, lucia.szalonova, roman.varga

December 2019

1 Pseudocode

Function findCorners(img, maxCor, thresh, dst)

The input values are img, maxCor, tresh, dst. Img expects input image, maxCor expects maximum number of corners. Tresh expects the selected threshold and dst the selected distance.

dy, dx = np.gradient(img)
Ixx = dx ^{2} Laplace oparator
Iyy = dy ^{2} Laplace oparator

An image gradient is a directional change in the intensity or color in an image. (dx,dy)

The second derivative (dx) in the x direction can be calculated in a discrete image as the difference between the brightness values side by side.

To detect isolated points can be used Laplace operator Δ^2.
Masks in the positive or negative variant are used for the Laplace operator

height = img.shape[0]
width = img.shape[1]

Height and width are representing the size of image. From the inserted image we get the size thanks to the shape function.

offset = 0

cornerList = []

The Corners List is an empty arrayList, to which we will adding the found corners.

In these two cycles, we look for the corners of the image.

for y in range(offset, height - offset):
for x in range(offset, width - offset):

Here we will calculate the squares

windowIxx = Ixx[y - offset:y + offset + 1, x - offset:x + offset + 1]

windowIyy = Iyy[y - offset:y + offset + 1, x - offset:x + offset + 1]

Here we will calculate the sum of squares

Sxx = Σ *windowIxx*

Syy = Σ *windowIyy*

If lambda1 (Sxx) and lambda2 (Syy) have large positive values, then a corner is found.

We are selecting the minimum of lambda1 (Sxx) and lambda2 (Syy)

r = min(Sxx, Syy)

If r is greater than tresh, we will add the corner to the end of the corner List.
if r is greater than thresh:

cornerList.append([x, y, r])

Our function will call the function filterCornersByDistance
and it will return filtered list of corners by euclidean distance comparism.

return filterCornersByDistance(cornerList, dst)[0:maxCor]