UFPB-CI-DSC
Disciplina: Visão Computacional
Prof. Augusto de Holanda B. M. Tavares
25 de Junho de 2024

12ª Atividade

Detecção de características de imagens

O código abaixo mostra como utilizar a biblioteca OpenCV para detectar os cantos de uma imagem utilizando os métodos de Harris e Shi-Tomasi e exibe os resultados.

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

%matplotlib inline

# Load image
filename = "chessboard.jpg"
img = cv.imread(filename)

# convert frame to grayscale and float32
gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
gray = np.float32(gray)

# get corners from harris method
dst1 = cv.cornerHarris(gray,2,3,0.04)
#result is dilated for marking the corners
kernel = np.ones((3, 3), "uint8")
dst1 = cv.dilate(dst1,kernel)

# Marks Harris corners in blue
# Threshold for an optimal value, it may vary depending on the image.
img[dst1>0.01*dst1.max()]=[0,0,255]

# get corners from shi-tomasi method
dst2 = cv.goodFeaturesToTrack(gray,25,0.01,10)
dst2 = np.intp(dst2)

# marks shi-tomasi corners in red
for i in dst2:
    x,y = i.ravel()
    cv.circle(img,(x,y),5,255,-1)

# create figure 1
```

```
fig1 = plt.figure(1)
ax1 = fig1.subplots()
ax1.imshow(img)
ax1.title.set_text('I')
ax1.set_xticks([])
ax1.set_yticks([])

# show figures
plt.show()
```

Utilizando o código acima como base e os conceitos apresentados em sala de aula, implemente um código que execute as seguintes tarefas:

1. Implemente um detector de *features* utilizando o método de Harris. Aplique o seu detector a uma imagem e compare os resultados com o método cv.cornerHarris().

2. Implemente um detector de *features* utilizando o método de Shi-Tomasi. Aplique o seu detector a uma imagem e compare os resultado com o método cv.goodFeaturesToTrack() disponível no OpenCV.

Funções de referência

Algumas funções de referência necessárias para implementar as tarefas acima são descritas abaixo:

```
corners = cv.cornerHarris(frame,blockSize,ksize,k)
```

- Retorna as *features* detectadas pelo método de Harris em um determinado $frame$, para um dado tamanho de bloco $blockSize$, abertura do operador de Sobel $ksize$ e parâmetro do algoritmo de Harris k.

```
corners = cv.goodFeaturesToTrack(frame,maxCorners,qualityLevels,minDistance)
```

- Retorna as *features* detectadas pelo método de Harris em um determinado $frame$, para um dado tamanho de bloco $blockSize$, abertura do operador de Sobel $ksize$ e parâmetro do algoritmo de Harris k.

```
grad = cv.Sobel(img,ddepth,dx,dy,ksize)
```

- Retorna o gradiente de Sobel de uma imagem. O parâmetro $ddepth$ define a profundidade de *pixels* da saída (em -1 é a mesma da imagem de entrada). dx e dy são as ordens da derivada em x e y, respectivamente. $ksize$ é o tamanho do *kernel* de Sobel utilizado, comumente definido em 3 ou em -1 (operador de Scharr).

```
eigvalues, _ = np.linalg.eig(m)
```

- Retorna os autovalores da matriz m.

```
cv.convertScaleAbs()
```

- Modifica a escala e devolve uma versão da entrada em valores absolutos de 8 *bits*.