Labo 5

Bert De Saffel

11 maart 2019

Oefening 10

Ik heb een functie `getDoGFilter(size, sigmabig, sigmasmall, angle)` gemaakt die de DogFilter zal aanmaken. Deze functie voert stapsgewijs de procedure van de opdracht uit.

Nieuwe functies

- **getGaussianKernel(ksize, sigma) → retval**

  Genereert een kolommatrix met *ksize* rijen. De matrix wordt gedefinieerd als

  $$G_i = \alpha * \exp\left(-\frac{(i-(ksize-1)/2)^2}{2*\sigma^2}\right)$$

- **getRotationMatrix2D(center, angle, scale) → retval**

  Genereert een 2×3 matrix van een twee-dimensionale rotatie. De *center* parameter stelt het rotatiepunt voor, de *angle* parameter de rotatiehoek en de *scale* parameter de schaal. De gegenereerde matrix heeft volgende vorm:

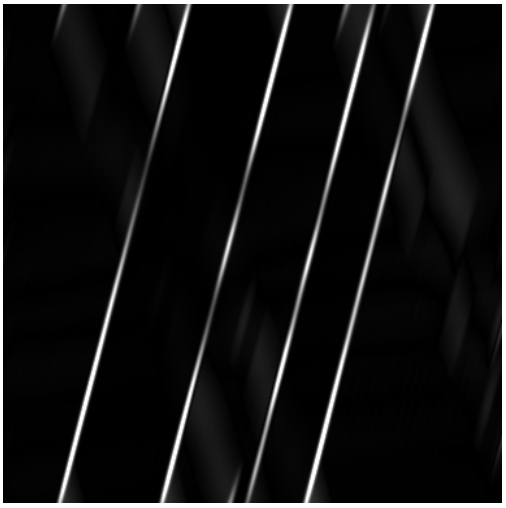
  $$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center\cdot x - \beta\cdot center\cdot y \\ -\beta & \alpha & \beta\cdot center\cdot x - (1-\alpha)\cdot center\cdot y \end{bmatrix}$$

  met

  $$\alpha = scale\cdot\cos(angle)$$
  $$\beta = scale\cdot\sin(angle)$$

Invoer en uitvoer

Figuur 1: Originele image.



Figuur 2: De randen van de gele strepen.

Oefening 11

De functie `detectLines(image, threshold1, threshold2)` → `dst` is een eigen wrapperfunctie die de lijnen zal detecteren. De randen worden bepaald met de Canny functie. De lijnen worden dan bepaald met de HoughLines functie. Het resultaat van HoughLines is een verzameling lijnen in de Houghruimte die elk gekarakteriseerd kunnen worden door ρ en θ. De vergelijking van zo een rechte is:

$$\rho = x\cos(\theta) + y\sin(\theta)$$

Om nu lijnen te kunnen tekenen op de figuur wordt eerst de vergelijking omgevormd in $y = ax + b$ vorm:

$$y = -x\cot(\theta) + \frac{\rho}{\sin(\theta)}$$

Hierbij is $a = -\cot(\theta)$ en $b = \rho/\sin(\theta)$. In Python kan een anonieme functie aangemaakt worden:

```
y = lambda x :  a*x + b
```

Als we nu voor elke lijn in de Houghruimte twee punten zoeken waarbij $x_0 = 1$ en $x_1 = width$ en $width$ de breedte is van de image, dan kunnen de twee punten in de beeldruimte gedefinieerd worden als:

```
point1 = (x0, y(x0))
point2 = (x1, y(x1))
```

Deze punten kunnen dan meegegeven worden aan de `line` functie van openCV. De lijnen zullen automatisch geclipped worden.

Nieuwe functies

- **Canny(image, threshold1, threshold2, ..., L2gradient) → edges**

  Deze functie zoekt randen in een figuur via het Canny algoritme. De hysteresis procedure vraagt om twee thresholds om pixels met een te lage waarde (threshold1) te verwerpen en die met een hoge waarde (threshold2) bij te houden.

- **HoughLines(image, rho, theta, threshold) → lines**

  Deze functie zoekt lijnen in een figuur. De parameters ρ en θ specificeren de resolutie van de Houghruimte.

- **line(img, pt1, pt2, color) → None**

  Tekent een lijnsegment op de opgegeven image. Deze functie verwacht de coördinaten van twee punten en RGB kleur van de lijn.

Invoer en uitvoer

Figuur 3: Originele image.

Figuur 4: Gedetecteerde lijnen.

Oefening 12

De wrapperfunctie `detectCorners(image)` voert het algoritme uit om de hoekpunten te bepalen. Deze functie maakt gebruik van `goodFeaturesToTrack()` om de coördinaten van de hoekpunten te bepalen. Deze hoekpunten worden dan omcirkeld via de functie `circle()`

Nieuwe functies

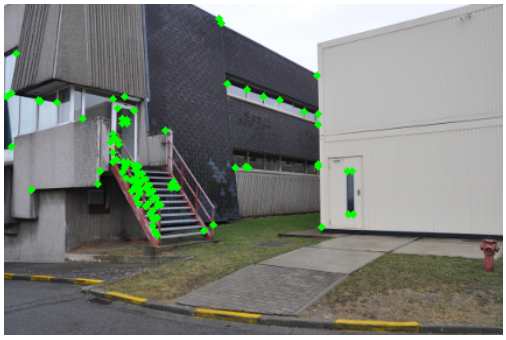
- **goodFeaturesToTrack(image, maxCorners, qualityLevel, minDistance) → corners**

  Deze functie bepaalt hoeken in een beeld. De *maxCorners* parameter bepaalt het aantal hoeken dat gevonden moet worden, de *qualityLevel* parameter zet een threshold op de waarde van de kwaliteit van een hoek en de *minDistance* parameter bepaalt minimum afstand tussen twee hoeken.

- **circle(img, center, radius, color) → None**

  Tekent een cirkel op de opgegeven image. De *center* parameter zijn de middelpuntcoördinaten van de cirkel, de *radius* parameter is de straal van de cirkel en de *color* parameter bevat de RGB kleur van de cirkel.

Invoer en uitvoer



Figuur 5: Hoekdetectie voor shots1.png.

Figuur 6: Hoekdetectie voor shots2.png.

Oefening 13

ORB (Oriented FAST and Rotated BRIEF) is ook een hoekdetector. Eerst worden de coördinaten en descriptoren van de punten opgehaald via een ORB object uit OpenCV. Met een BFMatcher (Brute Force matcher) object kunnen de descriptoren van de ORB features van twee verschillende figuren vergeleken worden. De methode `drawMatches` maakt verbindingen tussen punten die matchen.

Nieuwe functies

- **ORB_create(nfeatures) → orb**

  Creëert een ORB object waarmee $nfeatures$ features gedetecteerd mee kunnen worden. Het object heeft een methode `detectAndCompute(image, mask)` die de hoekpunten zal detecteren.

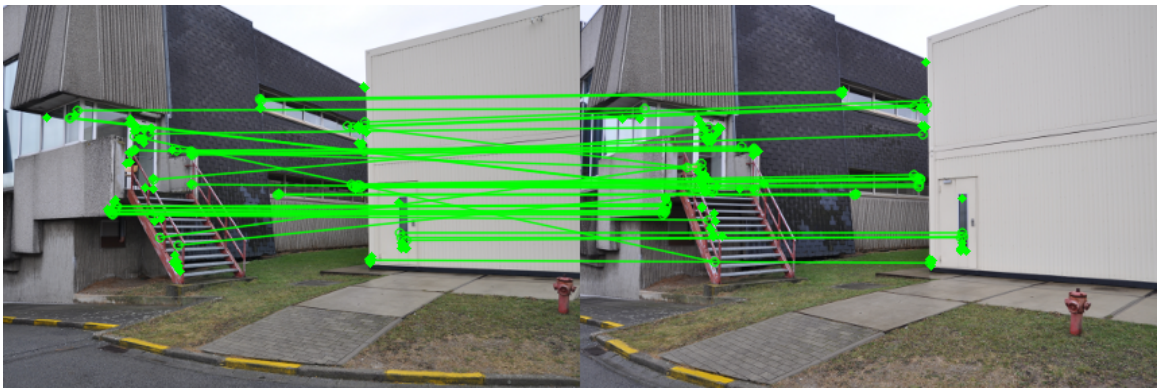
- **BFMatche_create(normType, crossCheck)**

  Deze functie creëert een BFMatcher object. Dit object bevat een methode `match(descriptors1, descriptors2)` die twee descriptors aanvaardt en een lijst teruggeeft van hoekpunten die matchen.

- **drawMatches**

  Deze functie zal de gedecteerde hoekpunten van twee figuren verbinden als er een match is.

Invoer en uitvoer



Figuur 7: Hoekmatches voor shots1.png en shots2.png.