eature Matching

Kategori: Computer Vision , 02 Şubat 2020 , JanFranco

emplate matching yazımızda bir resmi bir resimde aramıştık. Yani bir objeyi referans göstererek, obje takibi yapmıştık. Ancak mplate matching prensibindeki en büyük sorun, referans resmin, tıpatıp aynı olması gerekiyor. Feature matching'de ise böyle bir orunluluk yoktur. İleri seviye computer vision tekniklerindendir. OpenCV kütüphanesini kullanarak Python üzerinden görelim:

ütüphane ve resimlerimizi import edelim:

```
port cv2

real = cv2.imread('../data/reeses_puffs.png', 0)
reals = cv2.imread('../data/many_cereals.jpg', 0)

2.imshow('cereal', cereal)
2.waitKey(0)
2.imshow('cereals', cereals)
2.waitKey(0)
```

>

ature matching için birçok teknik mevcut. Burada göreceğimiz teknikler sadece syntax, method kullanımlarını öğrenmek için. Arkaplanda yapılan işlemler, kaplandaki matematik vs. gibi detaylar başka bir yazıda anlatılacaktır. İlk olarak Brute Force Detection with ORB Descriptors yöntemini görelim:

```
b = cv2.ORB_create()
1, des1 = orb.detectAndCompute(cereal, None)
2, des2 = orb.detectAndCompute(cereals, None)
 = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
tches = bf.match(des1, des2)
tches = sorted(matches, key=lambda x: x.distance)
realMatches = cv2.drawMatches(cereal, kp1, cereals, kp2, matches[:25], None, flags=2)
2.imshow('img', cerealMatches)
2.waitKey(0)
```

>

ute-Force Matching with SIFT Descriptors yöntemini görelim:

```
ft = cv2.xfeatures2d.SIFT_create()
1, des1 = sift.detectAndCompute(cereal, None)
2, des2 = sift.detectAndCompute(cereals, None)
 = cv2.BFMatcher()
tches = bf.knnMatch(des1, des2, k=2)
od = []
r match1, match2 in matches:
  if match1.distance < 0.75 * match2.distance:
      good.append([match1])
ftMatches = cv2.drawMatchesKnn(cereal, kp1, cereals, kp2, good, None, flags=2)
2.imshow('img', siftMatches)
2.waitKey(0)
```

>

ANN based Matcher

```
ANN_INDEX_KDTREE = 0
dex_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
arch_params = dict(checks=50)
ann = cv2.FlannBasedMatcher(index_params, search_params)
tches = flann.knnMatch(des1, des2, k=2)
od = []
r match1, match2 in matches:
  if match1.distance < 0.7 * match2.distance:
      good.append([match1])

annMatches = cv2.drawMatchesKnn(cereal, kp1, cereals, kp2, good, None, flags=0)
2.imshow('img', flannMatches)
2.waitKey(0)
```

>

nraki Yazı: Watershed Algorithm

orumlar

Henüz bir yorum bulunmuyor.

Yorum bırakın

İsim:

Yorum:

Ekle