بسم تعالی

## مقدمه:

بینایی پیشرفته ترین حس انسان است و تاثیر بسازیی در ادراک انسان دارد. پردازش تصاویر از جنبه های مختلف به ارتقای این ادراکات کمک می نماید عکس های که از منابع مختلف می آیند را می شود برای استفاده آداپته کرد. یا تصاویر را به نحو مناسبی ذخیره سازی کرد.یا با استخراج مفاهیم از تصاویر به درک کمک نمود.

زبان پایتون یک زبان برنامه نویسی مفسری و همه منظوره با سیستم نوع داینامیک است که توسط «گوییدو ونراسم» آغاز گشت این زبان اجازه می دهد تا منطق بیشتری در تعداد خطوط کمتر پیاده سازی شود. با اینکه نسبت به زبان های مانند c/c++ کند می باشد اما می شود پایتون را با c/c++ توصعه داد تا عملا کد کامپایل شده برای اجرا فراخوانی شود.

زبان و اکوسیستمی که پایتون فراهم می کند نه تنها برای پردازش تصویر بلکه بسیاری از فرآیند های علمی مناسب کتاب خانه هایی مانند PIL scikit-image و opencv-python نه تنها امکانات پردازش تصویر را فراهم می کنند بلکه این کار را به نحو قابل قبولی در سطح صنعتی انجام می دهند. پایتون برای توسعه و نمونه سازی سیستم های نرم افزاری به کار می رود بلکه می شود از امکانات فراهم شده ی آن در سیستم های نهفته نیز استفاده کرد.

کتابخانه python-opencv :

امکانات کتاب خانه opencv را در داخل قالبی که قابل انتقال و استفاده در پایتون است به صورت یک بسته گسترشی در ماژولی به نام Python-opencv نوشته شده. که قابل استفاده در پایتون دو و سه می باشد. در ادامه به تشریح استفاده از آن می پردازیم.

ماژول cv2 تبدیل کتابخانه opencv برای زبان پایتون است و numpy ابزار حساب ماتریسی پر قدرت و سریع برای پایتون

می باشد توابع ماژول cv2 تصاویر را به صورت ماتریس های چندبعدی numpy دریافت و بازگردانی می نمایند. کلاس numpy.ndarray آرایه های چند بعدی همگن.با کارایی و قدرت بالا پیاده سازی می کند نوع داده مربوطه (dtype) فرمت هر المان آرایه را توصیف می کند

نمونه ای از توابع ساخت آرایه های numpy در جدول زیر آمده است

|  |  |
| --- | --- |
| سازنده | توزیح |
| ndarray(shape, dtype=float, buffer=None, offset=0, strides=None, order=None), | آرایه چند بعدی همگن. نوع داده مربوطه (dtype) فرمت هر المان آرایه را توصیف می کند |
| zeros (shape, dtype=float, order='C) | آرایه با ابعاد shape از صفر ها |
| ones (shape, dtype=float, order='C) | آرایه با ابعاد shape از یک ها |
| linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None) | یک سری حسابی از مقدار start تا مقدار stop و با تعداد num را می سازد. |

## عملیات ورودی و خروجی

پردازش تصاویر دبجیتال شامل عملیاتی می شود که ورودی و خروجی آن­ها تصاویر هستند. پس نیاز است که بتوانیم این تصاویر را در حافظه باز گذاری کرده و نتیجه پردازش را در فایل ذخیره کنیم.

|  |  |
| --- | --- |
| تابع | توزیح |
| imread | یک تصویر را از فایل بار گذاری کرده و به صورت آرایه numpy.ndarray بر می گرداند در صورت شکست در خواندن فایل مقدار None برگردانده می شود |
| imwrite | یک آرایه را به صورت یک تصویر در یک فایل ذخیره می کند |
| imshow | یک تصویر را در پنجره مشخص شده نمایش می دهد. |

## جدول1-1 عملیات I/O با cv2

### عملیات ورودی و خروجی برای ویدیو

|  |  |  |
| --- | --- | --- |
| کلاس | توزیح | سازنده |
| VideoCapture | برای دریافت ویدیو یک شئ از این کلاس مورد نیاز است می توان شماره دستگاه یا مسیر فایل ویدیو را به سازنده ارسال کرد. | VideoCapture(0),  VideoCapture(‘C:/users/me/  videos/my\_video’) |
| VideoWriter | برای ذخیره فریم به فریم در یک فایل ویدیو استفاده می شود. به سازنده باید مسیر فایل و یک شئ VideoWriter\_fourCC داده شود. تعداد فریم در ثانیه و ابعاد نیز می تواند توسط کاربر مشخص شود. | cv2.VideoWriter('output.avi', fourcc, fps=20.0, frame\_size=(640,480)) |
| VideoWriter\_fourcc | کد چهار رقمی کدک (فرمت ذخیره سازی) را کپسوله سازی می کند. | cv2.VideoWriter\_fourcc('M','J','P','G'),  cv2.VideoWriter\_fourcc('MJPG’) |

جدول 1-2 کلاس­هایی که برای خواندن و نوشتن ویدیو در ورودی و خروجی استفاده می­شوند.

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| read | در صورت وجود فریمی برای خواندن و باز بودن VideoCapture یک فریم ویدیو دریافت شده را بر می گرداند. | cap = cv2.VideoCapture(0)  ret, frame = cap.read() |
| write | در صورت باز بودن VideoWriter فریم تصویر را در آن ذخیره می کند. | out = cv2.VideoWriter('output.avi', fourcc, fps=20.0, frame\_size=(640,480)  out.write(frame) |
| isOpened | در صورت وجود باز بودن مقدار True را بر می گرداند. | if cap.isOpened():  #todo |
| release | فایل یا دستگاه ورودی یا خروجی را آزاد می کند. | cap.release() |

## عملیات پایه

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| addWeighted | جمع وزن دار دو تصویر را بر می گرداند | cv2.addWeighted(img1,0.7,img2,0.3,0) |
| bitwise\_not | یک تصویر دودویی را معکوس می کند. | cv2.bitwise\_not(mask) |
| bitwise\_and | دو تصویر دریافت کرده و با هم و پیکسل به پیکسل ترکیب عطفی می کند اگر ماسک به ازای یک پیکسل یک باشد آن پیکسل صفر می شود. | cv2.bitwise\_and(src1, src2, mask = mask) |
| bitwise\_or | دو تصویر دریافت کرده و با هم و پیکسل به پیکسل ترکیب فصلی می کند اگر ماسک به ازای یک پیکسل یک باشد آن پیکسل صفر می شود. | cv2.bitwise\_or(src1, src2, mask = mask) |





### تبدیل رنگ در opencv

تبدیل فضای رنگ در cv2 با استفاده از تابع cvtColorانجام می شود امضای این تابع به صورت زیر است

cvtColor(src, code[, dst[, dstCn]]) -> dst

src تصویر مبدا است و dst همان تصویر در فضای رنگی جدید code نماینده کد تبدیل فضای رنگ است که حالات بسیاری دارد. جدول زیر شمال چند نمونه می شود

|  |  |
| --- | --- |
| کد تبدیل رنگ | توزیح |
| COLOR\_RGB2GRAY | تبدیل cvtColor از فضای BGR به خاکستری انجام گیرد |
| COLOR\_BGR2YCrCb | تبدیل cvtColor از فضای BGR به YCrVb انجام گیرد |
| COLOR\_BGR2HSV | تبدیل cvtColor از فضای BGR به HSV انجام گیرد |



### آستانه گذاری

### آستانه گذاری ساده

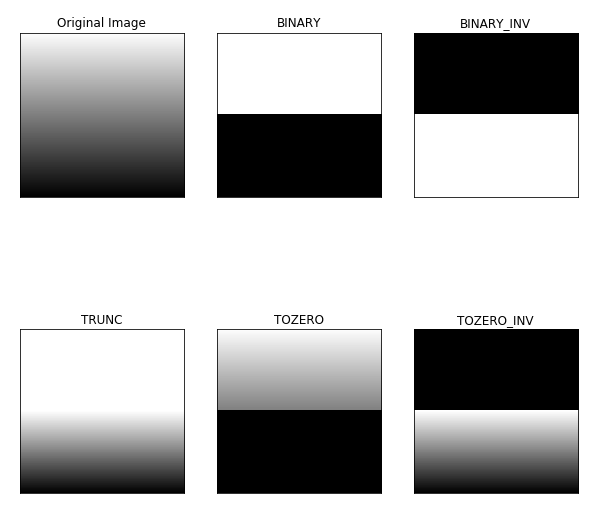
تابع cv2.threshold با امضای زیر برای آستانه گذاری ساده با استفاده از یک مقدار آستانه برای تمام تصویر استفاده می شود

threshold(src, thresh, maxval, type[, dst]) -> retval, dst

src تصویر اولیه thresh مقدار آستانه و maxval سقف مقدار قابل قبول است dst تصویر آستانه گذاری شده است چند نمونه از مقادیر قابل قبول برای type در زیر آمده که نحوه رفتار با مقادیر بین آستانه و حداکثر را تائین می کند در صورت استفاده از type = *cv2.THRESH\_OTSU* retval مقدار آستانه بهینه یافت شده را بر می گرداند وگرنه همان مقدار thresh را خواهد داشت.

|  |
| --- |
| cv2.THRESH\_BINARY |
| cv2.THRESH\_BINARY\_INV |
| cv2.THRESH\_TRUNC |
| cv2.THRESH\_TOZERO |
| cv2.THRESH\_TOZERO\_INV |

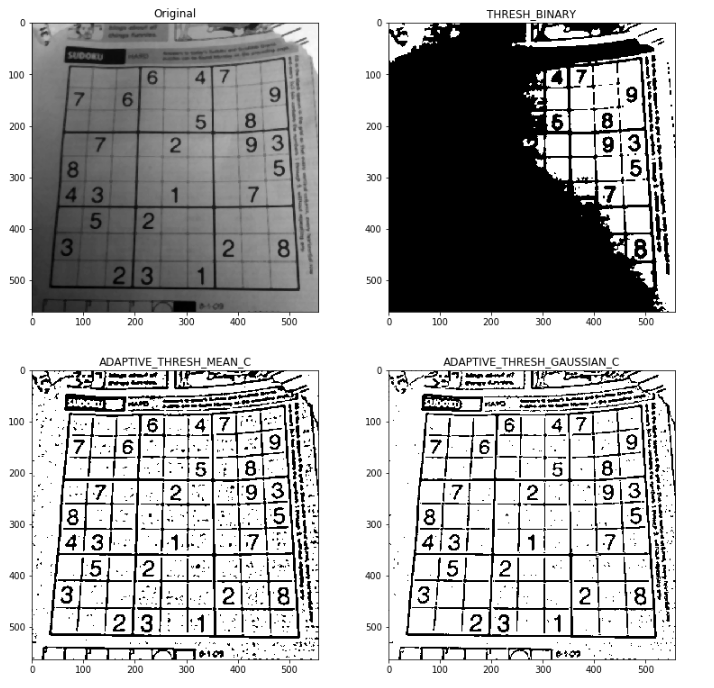




### آستانه گذاری تطبیقی

adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) -> dst

|  |  |
| --- | --- |
| متد تطبیق | توزیح |
| ADAPTIVE\_THRESH\_MEAN\_C | استفاده از میانه برای آستانه گذاری |
| ADAPTIVE\_THRESH\_GAUSSIAN\_C | استفاده از میانه وزن دار گوسی برای آستانه گذاری |



### تبدیلات هندسی

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | فراخوانی |
| resize | برای تغییر اندازه عکس استفاده می شود ورودی interpolation می­تواند برای مشخص کردن نحوه برون-یابی استفاده شود | resize(src, dsize[, dst[, fx[, fy[, interpolation]]]]) -> dst |
| warpAffine | تصویر ورودی را با استفاده از ماتریس M دچار تبدیل خطی می کند | warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst |

تجانس

با استفاده از تابع Cv2.resize انجام می شود

نگاشت

با استفاده از cv2.warpAffine و دادن تعیین ماتریس جابجایی در مقدار M انجام می شود

دوران

برای ساختن یک ماتریس تبدیل که تنها عمل دوران را انجام دهد می­توان از تابع زیر استفاده کرد ورودی شامل یک تابل که محور دوران را مشخص می کند و مقدار زاویه دوران می باشد.

M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)

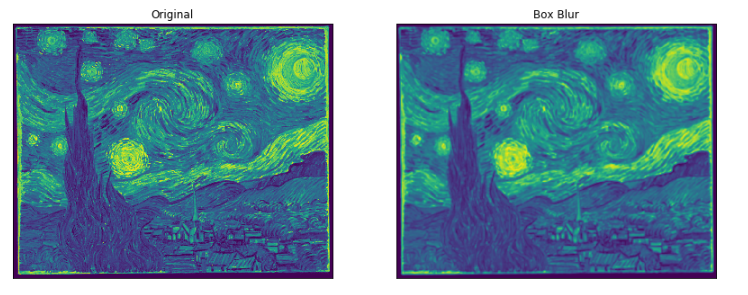
سپس با استفاده از تابع warpAffine و ماتریس دوران تبدیل را انجام داد.

dst = cv2.warpAffine(img,M,(cols,rows)

### صاف کردن و کاهش نویز

میانگین گیری:

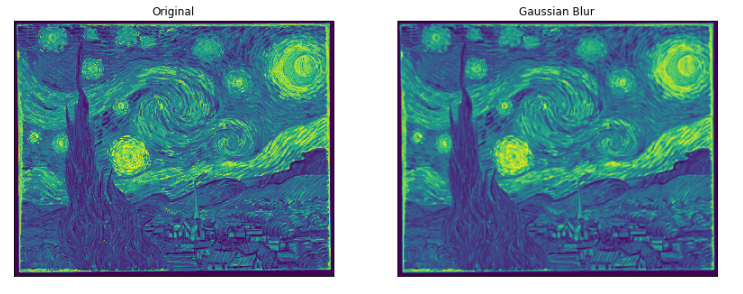




میانگین گیری جعبه ای با کرنل در ابعاد 5 در 5

فیلتر گوسی



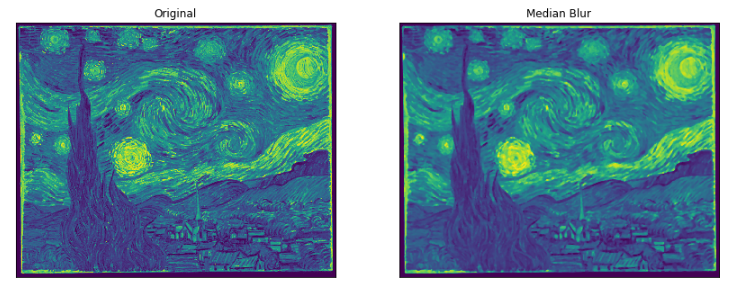


استفاده از یک کرنل گوسی 5 در 5 با مقدار سیگما صفر

cv2.GaussianBlur(img,(5,5),0)

### فیلتر میانه



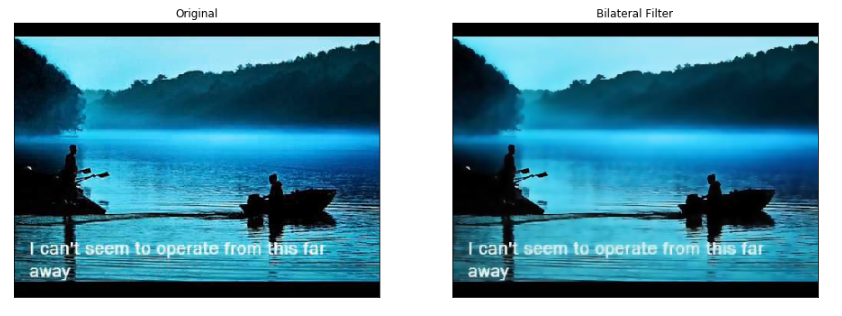


فیاتر میانه 5 در 5

cv2.medianBlur(img,5)

فیلتر دو طرفه





فیلتر دو طرفه با شعاع 9 و مقادیر سیگما 75

cv2.bilateralFilter(img,9,75,75)

### تغییرات ریخت شناسی

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | فراخوانی |
| erode | تصویر مبدا را با استفاده از المان ساختاری فرسایش می دهد. | erode(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst |
| dilate | تصویر مبدا را با استفاده از المان ساختاری منحطط می کند. | dilate(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst |
| morphologyEx | با استفاده از انحطاط و فرسایش عملیات پیچیده ریخت شناختی را انجام می دهد که نوع عمل باید با پرچم مربوطه ذکر شود | morphologyEx(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst |

مقادیر ورودی op برای تابع morphologyEx

|  |  |
| --- | --- |
| **OP** |  |
| cv2.MORPH\_OPEN | افتتاح |
| MORPH\_CLOSE | بستن |
| MORPH\_GRADIENT | شیب ریختی |
| MORPH\_TOPHAT | کلاه بالا |
| MORPH\_BLACKHAT | کلاه پایین |

### گرادیان تصویر

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| Sobel | مشتق سوبل یک تصویر را به دست می آورد. | Sobel(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[, borderType]]]]]) -> dst |
| Laplacian | لاپلاسین تصویر را با استفاده از مشتق سوبل محاسبه می کند | Laplacian(src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]]]) -> dst |

### لبه یاب کنی

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | فراخوانی |
| Canny | پیاده سازی الگوریتم کنی در opencv | Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]]) -> edges |

### کونتر

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | فراخوانی |
| findContours | کنتور های مئجود در تصویر را با استفاده از الگوریتم suzuki89 می یابد | findContours(image, mode, method[, contours[, hierarchy[, offset]]]) -> image, contours, hierarchy |
| drawContours | یک ماتریس از کنتورهای به دست آمده را گرفته و بر روی تصویر اصلی اضافه می کند | drawContours(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]]) -> image |

|  |  |
| --- | --- |
| مقادیر ممکن برای ورودی method |  |
| CHAIN\_APPROX\_NONE | تمام نقاط کنتور را نگه می دارد. |
| CHAIN\_APPROX\_SIMPLE | نقاط اضافی کنتور را حذف کرده و فشرده می سازد. |

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| moments | یک دیکشنری از گشتاورهای تا درجه سوم تصویر را بر­می گرداند | moments(array[, binaryImage]) -> retval |
| contourArea | گشتاور درجه صفر از کنتور را بر می گرداند. | contourArea(contour[, oriented]) -> retval |
| arcLength | مقدار طول محیط کنتور را بر می گرداند | arcLength(curve, closed) -> retval |
| approxPolyDP | یک خم را با استفاده از خم دیگر با خطای کمتر از epsilon تقریب می زند. | approxPolyDP(curve, epsilon, closed[, approxCurve]) -> approxCurve |
| convexHull | کانوکس هال یک بردار از نقاط را می یابد | convexHull(points[, hull[, clockwise[, returnPoints]]]) -> hull |
| isContourConvex | اگر یک کنتور محدب باشد مقدار True وگرنه False را بر می گرداند | isContourConvex(contour) -> retval |
| minAreaRect | کوچکترین مستطیل زاویه دار که حاوی یک مجموعه نقاط در دو بعد باشد را بر می گرداند | minAreaRect(points) -> retval |
| minEnclosingCircle | کوچکترین دایره ای که حاوی یک مجموعه نقاط در دو بعد باشد را بر می گرداند | minEnclosingCircle(points) -> center, radius |
| fitEllipse | کوچکترین لوزی زاویه دار که حاوی یک مجموعه نقاط در دو بعد باشد را بر می گرداند | fitEllipse(points) -> retval |
| fitLine | بهترین خط تقریب برای چند نقطه را بر می گرداند | fitLine(points, distType, param, reps, aeps[, line]) -> line |
| boundingRect | مستطیل قائم حاوی نقاط را برمی گرداند. | boundingRect(points) -> retval |

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| convexityDefects | انحراف از تحدب کنتور را می یابد | convexityDefects(contour, convexhull[, convexityDefects]) -> convexityDefects |
| pointPolygonTest | مشخص می کند که آیا یک نطقه داخل کنتور روی آن یا خارج از آن است | pointPolygonTest(contour, pt, measureDist) -> retval |
| matchShapes | با استفاده از گشتاور های هیو دو شئ را مقایسه می کند | matchShapes(contour1, contour2, method, parameter) -> retval |

### هیستوگرام

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| calcHist | برای محاسبه هیستوگرام یک بعدی | calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]]) -> hist |
| equalizeHist | عملیات هموارسازی هیستوگرام را بر روی تصویر انجام می دهد | equalizeHist(src[, dst]) -> dst |

### تبدیل فوریه

|  |  |  |
| --- | --- | --- |
| تابع | توزیح | مثال |
| dft | انجام عمل تبدیل فوریه گسسته بر آرایه یک یا دو بعدی | dft(src[, dst[, flags[, nonzeroRows]]]) -> dst |
| idft | انجام عکس عمل تبدیل فوریه گسسته بر آرایه یک یا دو بعدی | idft(src[, dst[, flags[, nonzeroRows]]]) -> dst |

### تطبیق الگو

matchTemplate

matchTemplate(image, templ, method[, result[, mask]]) -> result

'cv2.TM\_CCOEFF', 'cv2.TM\_CCOEFF\_NORMED', 'cv2.TM\_CCORR',

'cv2.TM\_CCORR\_NORMED', 'cv2.TM\_SQDIFF', 'cv2.TM\_SQDIFF\_NORMED'

### تبدیل خطی هاف

HoughLines

HoughLines(image, rho, theta, threshold[, lines[, srn[, stn[, min\_theta[, max\_theta]]]]]) -> lines

### تبدیل دایره ای هاف

HoughCircles

HoughCircles(image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]]) -> circles

### تقسیم بندی تصویر با الگوریتم آب فشان

watershed

watershed(image, markers) -> markers

کنج یاب هریس

cornerHarris

cornerHarris(src, blockSize, ksize, k[, dst[, borderType]]) -> dst

کنج یاب شی­توماسی

goodFeaturesToTrack

goodFeaturesToTrack(image, maxCorners, qualityLevel, minDistance[, corners[, mask[, blockSize[, useHarrisDetector[, k]]]]]) -> corners

تبدیل ویژگی های بدون تغییر با مقیاس(SIFT)

sift = cv2.SIFT()

kp = sift.detect(gray,None)

img=cv2.drawKeypoints(gray,kp)

compute

detectAndCompute

img=cv2.drawKeypoints(gray,kp)

ویژگی های قدرتمند تسریع شده(SURF)

surf = cv2.SURF(400)

# Find keypoints and descriptors directly

>>> kp, des = surf.detectAndCompute(img,None)

الگوریتم FAST

fast = cv2.FastFeatureDetector()

# find and draw the keypoints

kp = fast.detect(img,None)

img2 = cv2.drawKeypoints(img, kp, color=(255,0,0))

BRIEF

import numpy as np

import cv2

from matplotlib import pyplot as plt

img = cv2.imread('simple.jpg',0)

# Initiate STAR detector

star = cv2.FeatureDetector\_create("STAR")

# Initiate BRIEF extractor

brief = cv2.DescriptorExtractor\_create("BRIEF")

# find the keypoints with STAR

kp = star.detect(img,None)

# compute the descriptors with BRIEF

kp, des = brief.compute(img, kp)

print brief.getInt('bytes')

print des.shape

ORB

import numpy as np

import cv2

from matplotlib import pyplot as plt

img = cv2.imread('simple.jpg',0)

# Initiate STAR detector

orb = cv2.ORB()

# find the keypoints with ORB

kp = orb.detect(img,None)

# compute the descriptors with ORB

kp, des = orb.compute(img, kp)

# draw only keypoints location,not size and orientation

img2 = cv2.drawKeypoints(img,kp,color=(0,255,0), flags=0)

plt.imshow(img2),plt.show()

### Feature Matching

bf = cv2.BFMatcher(cv2.NORM\_HAMMING, crossCheck=True)

# Match descriptors.

matches = bf.match(des1,des2)

# Sort them in the order of their distance.

matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.

img3 = cv2.drawMatches(img1,kp1,img2,kp2,matches[:10], flags=2)

import numpy as np

import cv2

from matplotlib import pyplot as plt

MIN\_MATCH\_COUNT = 10

img1 = cv2.imread('box.png',0) # queryImage

img2 = cv2.imread('box\_in\_scene.png',0) # trainImage

# Initiate SIFT detector

sift = cv2.SIFT()

# find the keypoints and descriptors with SIFT

kp1, des1 = sift.detectAndCompute(img1,None)

kp2, des2 = sift.detectAndCompute(img2,None)

FLANN\_INDEX\_KDTREE = 0

index\_params = dict(algorithm = FLANN\_INDEX\_KDTREE, trees = 5)

search\_params = dict(checks = 50)

flann = cv2.FlannBasedMatcher(index\_params, search\_params)

matches = flann.knnMatch(des1,des2,k=2)

# store all the good matches as per Lowe's ratio test.

good = []

for m,n in matches:

if m.distance < 0.7\*n.distance:

good.append(m)

تحلیل ویدیو

Meanshift

meanShift

Camshift

camShift

Lucas-Kanade Optical Flow in OpenCV

Dense Optical Flow in OpenCV

calcOpticalFlowPyrLK

Background Subtraction

calcOpticalFlowFarneback

BackgroundSubtractorMOG

createBackgroundSubtractorMOG

BackgroundSubtractorMOG2

createBackgroundSubtractorMOG2

BackgroundSubtractorGMG

createBackgroundSubtractorGMG

|  |  |
| --- | --- |
| ماژول | توزیح |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| کلاس | توزیح | سازنده |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| تابع | توزیح | مثال | خروجی |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |