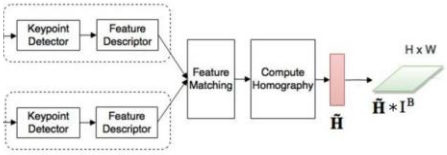
Features

فهیم جعفری

اطلاعات گزارش	چکیده
تاریخ:	استخراج ویژگی های تصاویر یکی از عملیات مهم در پردازش تصویر است که در مقایسه تصاویر بر اساس محتوا مورد استفاده قرار می گیرد. در بینائی ماشین و پردازش تصویر با استفاده از بعضی عملیات ریاضی نظیر تشخیص لبه بوسیله گرادیان و یا اعمال فیلترهای مناسب ویژگی-های تصویر نظیر لبه ها، خطوط، انحناها، گوشه ها و مرزها را می‌توان استخراج کرد. استخراج این ویژگی ها، نمایش و تحلیل صحنه های تصویر را آسان تر می سازد. در روش های موجود برای استخراج کلیه الگوهای ویژگی ها باید جستجو و شناسائی گردندHarris Corner Detector یک عملگر تشخیص گوشه است که معمولاً در الگوریتم های دید رایانه ای برای استخراج گوشه ها و ویژگی های استنباط یک تصویر استفاده می شود.. در مقایسه با مورد قبلی ، ردیاب گوشه هریس به جای استفاده از تکه های جابجایی برای هر زاویه 45 درجه ، دیفرانسیل نمره گوشه را با توجه به جهت مستقیم در نظر می گیرد و در تشخیص بین لبه ها و گوشه ها دقیق تر به اثبات رسیده است. .
واژگان کلیدی: Feature Sift Surf Descriptor Transformation Harris Non maximal suppression Estimate geometry	

1-مقدمه

2-شرح تکنیکال



- **Estimate geometry**

از اوایل دهه 2000 ، ثبت تصاویر بیشتر از رویکردهای سنتی مبتنی بر ویژگی استفاده می کرده است. این رویکردها بر اساس سه مرحله انجام می شود: شناسایی کلید واژه ها و توضیحات ویژگی ها ، تطبیق ویژگی ها و تاب آوردن تصویر. به طور خلاصه ، ما نقاط مورد علاقه در هر دو تصویر را انتخاب می کنیم ، هر نقطه مورد نظر در تصویر مرجع را با معادل آن در تصویر سنجیده مرتبط می کنیم و تصویر حس شده را طوری تغییر می دهیم که هر دو تصویر تراز شوند.

تعین ویژگی و تعریف descriptor

نکته کلیدی یک نکته جالب است. آن را در یک تصویر (گوشه ها ، لبه ها ، و غیره) مهم و متمایز تعریف می کند. وکتور ویژگی های حاوی ویژگی های اساسی کلمات کلیدی: یک توصیف کننده باید در برابر تحولات تصویر (بومی سازی ، مقیاس ، روشنایی و غیره) مقاوم باشد. بسیاری از الگوریتم ها تشخیص کلید و توضیحات ویژگی را انجام می دهند:

SIFT (تبدیل ویژگی متغیر متغیر) الگوریتم اصلی است که برای تشخیص کلید واژه استفاده

می شود اما برای استفاده تجاری رایگان نیست.
توصیف کننده ویژگی SIFT در مقیاس بندی یکنواخت ، جهت گیری ، تغییر روشنایی و تا حدی غیرمستقیم برای ایجاد اعوجاج ثابت نیست.

SURF (Speeded Up Robust Features) یک ردیاب و توصیف کننده است که از SIFT الهام گرفته است. این مزیت از چندین برابر سریع تر ارائه می دهد. همچنین ثبت اختراع شده است.

تطبیق ویژگی :

پس از مشخص شدن کلیدهای کلیدی در هر دو تصویر که یک زوج را تشکیل می دهند ، ما باید نقاط کلیدی هر دو تصویر را که در واقعیت با همان نقطه مطابقت دارند ، با هم هماهنگ کنیم یا "مطابقت" پیدا کنیم. یک روش ممکن BFMatcher.knnMatch است. این تطابق فاصله بین هر جفت توصیف کننده کلید را اندازه گیری می کند و برای هر نقطه کلیدی بهترین k خود را با حداقل فاصله بازمی گرداند

سپس یک ترشولد بر روی فاصله بین ویژگی ها گرفته میشود و بهترین فاصله ها (کمترین) را انتخاب میکنیم

پیدا کردن ماتریس transformation :

پس از تطبیق حداقل چهار جفت کلید ، می توانیم یک تصویر را نسبتاً به تصویر دیگر تبدیل کنیم. به این شکل تصویر پیچشی گفته می شود. هر دو تصویر از یک سطح مسطح یکسان در فضا توسط یک هموگرافی مرتبط است. هوموگرافی ها تحولات هندسی هستند که دارای 8 پارامتر رایگان هستند و توسط یک ماتریس x33 نشان داده شده اند. آنها نمایانگر هرگونه اعوجاج ساخته شده به یک تصویر در کل (بر خلاف تغییر شکل های محلی) هستند. بنابراین ، برای به دست آوردن تصویر حس شده تبدیل شده ، ما ماتریس هموگرافی را محاسبه می کنیم و آن را بر روی تصویر سنجیده اعمال می کنیم

- **Corner detection**

الگوریتم تشخیص گوشه هریس همچنین با نام Hectoris & Stephens detector یکی از ساده ترین آشکارسازهای گوشه ای موجود است. ایده این است که نقاط مورد علاقه محلی را که محله اطراف آن در بیش از یک جهت قرار دارد نشان دهد. ایده اصلی الگوریتم یافتن اختلاف شدت برای جابجایی (u ، v) در کلیه جهات است که به شرح زیر است:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)]^2}_{\text{intensity}}$$

عملکرد پنجره یا یک پنجره مستطیلی یا یک پنجره گاوسی است که وزن پیکسل ها را در (x، y) می دهد. معادله فوق را می توان بیشتر با استفاده از گسترش تیلور تقسیم کرد که فرمول نهایی را به ما می دهد

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

که

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

Ix و Iy به ترتیب مشتقات تصویر در جهت x و y هستند. می توان با استفاده از هسته sobel ، مشتق را محاسبه کرد

سپس ما در نهایت پاسخ هریس R داده شده توسط:

$$R = det(M) - k(trace(M))^2$$

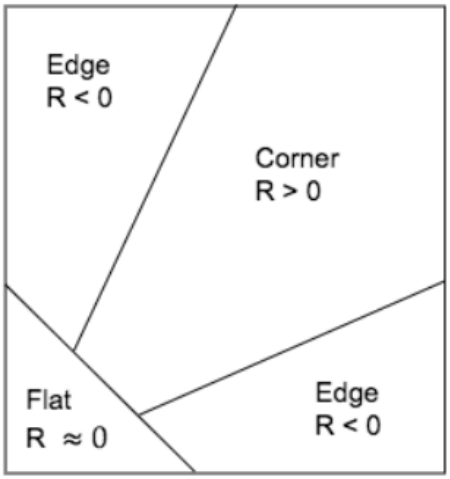
که

$$A = I_x I_x \oplus w$$
$$B = I_y I_y \oplus w$$
$$C = I_x I_y \oplus w$$

$$det(M) = AB - C^2 \; or \; \lambda_1 \lambda_2$$
$$trace(M) = A + B \; or \; \lambda_1 + \lambda_2$$

که در آن A، B و C نوارهای پنجره تعریف شده توسط w هستند. لامبداها مقادیر Eigen از M هستند.

گوشه ها را با استفاده از مقدار R می یابیم.



فرایند الگوریتم تشخیص هریس گوشه

1. تصویر رنگی تبدیل به Grayscale
2. محاسبه مشتق در جهت x ,y
3. محاسبه پاسخ هریس
4. ترشولد
5. لبه ها و گوشه ها را با استفاده از R پیدا کنید

Non maximal suppression:

در این روش تصویری که در ان فقط گوشه ها میباشد را به بلوک هایی تقسیم میکنیم و گوشه هایی که در ان بلوک ماکسیمم و بزرگتر از ترشولد است را انتخاب میکنیم که در واقع این پنجره را به عکسی که فقط گوشه ها دران است کانوالو میشود و نقاط مطلوب را انتخاب میکنیم

3-شرح نتایج

- **Estimate geometry**

تصویر رنگی اصلی

تصویر رفرنس

جدول مربوط به گزارش ssim,mmse,mp

	ssim	mmse	mp
1.bmp	0.9080400105353 8	56.3538932800293	162
2.bmp	0.9078572647136668	44.32229995727539	192
3.bmp	0.9433251507926815	122.48328143037396	174
4.bmp	0.8445714323454753	121.75693130493164	97
original.bmp	0.9633798166716 35	10.523101806640625	400
mean	0.9134347350117678	71.08790155585018	205.0
stdev	0.0452501762946 2466	49.52377975686869	114.74754899343166

تصویر زیر تصویری است که ماتریس تبدیل ان از تصویر چرخش یافته و هیستوگرام متعادل شده تصویر اصلی و تصویر رفرنس بدست امده و این ماتریس بر روی تصویر فقط چرخش یافته در پوشه 2 اعمال شده است میباشد که برای نقاطی که در اثر چرخش از بین رفته اند درونیابی شده است

با توجه به اینکه بر روی تصویر متعادل سازی هیستوگرام اعمال شده است و اینکه الگوریتم surf نسبت به تغییرات illumination ضعیف است باید تعداد ویژگی های انطباق داده شته کاهش یابد که این نتیجه را نیز در جدول میبینیم که تعداد ویژگی ها کاهش یافته است و همینطور از لحاظ ساختاری نیز شبیه تصویر اصلی میباشد مشاهده میشود که تصویر تا حد خوبی بازسازی شده است در این بازسازی از ترشولد .75 با روشی که در مقاله اقای دیوید لو میباشد استفاده شده است که به صورت تجربی خودم بدست اوردم و تعداد ویژگی هایی که انطباق داده شد 162 میباشد که تعداد خوبی است در حالت عادی یعنی خود تصویر اورجینال تعداد ویژگی هایی که انطباق داده میشود برابر با 400 میباشد که در جدول نیز گزارش شده است پس در نتیجه با اعمال متعادل سازی هیستوگرام و چرخش تقریبا 60 درصد ویژگی هایی که انطباق داده شده است را از دست داده ایم

تصویر زیر تصویر بازسازی شده از تصویر 2 فولدر attack میباشد

با توجه به اینکه الگوریتم surf نسبت به بلور و شارپ قوی و مقاوم میباشد انتظار میرود که تعداد ویژگی های انطباق داده شده کاهش پیدا نکند که همانطور نیز در جدول مشاهده میشود که ترشولد برابر با .75 و تعداد ویژگیهایی که انطباق داده شد تا ماتریس تبدیل ساخته شود برابر با 192 میباشد مشاهده میشود که تعداد انطباق ویژگی ها افزایش یافته است پس نسبت به شارپ کردن و کشیدگی قوی میباشد و همینطور mmse نیز کاهش یافته است که نشان میدهد تصویر کمتر نسبت به حالت قبل خراب شده است این نتایجی که گرفته میشود از روی جدولی میباشد که در پایین اورده شده است ولی مقدار ssim خوب است و از لحاظ ساختاری شبیه تصویر اصلی میباشد

تصویر زیر مربوط به تصویر 3 فولدر attack میباشد

با توجه به این که تصویر بلور و کراپ شده است ولی باز هم تعداد ویژگی های که انطباق داده شده است برابر با 174 میباشد و تصویر به خوبی بازسازی شده است پس الگوریتم surf نسبت به بلور کردن مقاوم میباشد و توانسته ویژگی ها را تشخیص دهد ولی مقدار mmse افزایش یافته است و این به معنی است که تصویر خراب شده است ولی مقدار ssim ان خیلی خوب است و این بدان معنی است که از لحاظ ساختاری به تصویر اصلی خیلی شبیه است

تصویر زیر مربوط به تصویر 4 فولدر attack میباشد

با توجه به اینکه تصویر کشیده شده است و فیلتر bilatfilt اعمال شده است تعداد ویژگی های که انطباق داده شده است نیز به 97 کاهش یافته است پس نتیجه میگیریم که الگوریتم sift نسبت به کشیدگی ضعیف میباشد . مقاومتی ندارد که کاهش مقدار ssim نیز بر این موضوع تاکید میکند

- **Corner detection**

در تصویر زیر الگوریتم هریس گوشه های تصویر را پیدا کرده است

زوم شده بر روی قسمتی از ان

مشاهده میشود که تمام گوشه ها را به خوبی پیدا نکرده است تعداد گوشه هایی که در این تصویر پیدا کرده است برابر است با 152320 ولی درحالی که گوشه های نزدیک را به هم وصل کرده است برای همین ترشولد را کم میکنیم شاید بهتر شود و دیگر گوشه ها به هم نچسبد

عیب این ترشولد این است که بعضی از گوشه ها رو تشخیص نداده است برای همین از روش non maximal suppersion استفاده میکنیم که تصویر که در ان فقط گوشه ها است را به بلاک هایی تقسیم میکند و در صورتی که در ان بلاک بیشترین مقدار را داشته باشد ان را به عنوان گوشه در نظر میگیرم که خروجی ان به شکل زیر است

مشاهده میشود که گوشه ها به خوبی پیدا شده است

اما عیب روش هریس این است که نسبت به تغییر اندازه تصویر یا اسکیل ضعیف است در تصویر زیر که تصویر کوچک شده است مشاهده میشود که گوشه ها به خوبی تشخیص داده نشده است

جدول زیر تعداد گوشه های پیدا شده در تصویر با اسکیل های مختلف در روش هریس معمولی و non maximal suppersion نشان میدهد

	harris	non maximal suppersion
1	152320	8600
2	51128	2814
3	30388	1593
4	20289	1071

مشاهده میشود که با کوچک شدن اسکیل تصویر تعداد گوشه های تشخیص داده شده کاهش یافته است

4-کد برنامه

کد قسمت 7.1

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from skimage.metrics import
structural_similarity as
compare_ssim
from skimage.metrics import
mean_squared_error
import statistics


def run():
    img_names =
['1.bmp','2.bmp','3.bmp','4.bmp','or
iginal.bmp']
    org_image =
cv2.imread('../image/Original.bmp',c
v2.IMREAD_GRAYSCALE)
    ref_image      =
cv2.imread('../image/Reference.bmp',
cv2.IMREAD_GRAYSCALE)
    data    = []
    for img_name in img_names:
        img_attack1     =
cv2.imread('../image/Attack
1/{}'.format(img_name),cv2.IMREAD_GR
AYSCALE)
        img_attack2     =
cv2.imread('../image/Attack
2/{}'.format(img_name),cv2.IMREAD_GR
AYSCALE)
        surf     =
cv2.xfeatures2d.SURF_create(400)
        kp1, des1 =
surf.detectAndCompute(ref_image,None
)
        kp2, des2 =
surf.detectAndCompute(img_attack1,No
ne)

        # BFMatcher with default
params
        bf      = cv2.BFMatcher()
        matches =
bf.knnMatch(des1,des2,k=2)
        good_matches = []
        for m,n in matches:
            if m.distance <
0.75*n.distance:

good_matches.append([m])
        # good_matches = matches
        # Select good matched
keypoints
        ref_matched_kpts    =
np.float32([kp1[m[0].queryIdx].pt
for m in good_matches])
        sensed_matched_kpts =
np.float32([kp2[m[0].trainIdx].pt
for m in good_matches])

        # Compute homography
        H, status    =
cv2.findHomography(sensed_matched_kp
ts, ref_matched_kpts,
cv2.RANSAC,5.0)
        print(H)
        # Warp image
        warped_image =
cv2.warpPerspective(img_attack2, H,
(ref_image.shape[1],
ref_image.shape[0]))
        if img_name == '3.bmp':
            cropfrom = 63
            cropto =
warped_image.shape[0] - cropfrom

cv2.imwrite('../image/hw7.1/Attack-
warped{}.jpg'.format(img_name),warpe
d_image[cropfrom:cropto,cropfrom:cro
pto])

cv2.imwrite('../image/hw7.1/Attack-
diff{}.jpg'.format(img_name),org_ima
ge[cropfrom:cropto,cropfrom:cropto]
-
warped_image[cropfrom:cropto,cropfro
m:cropto])
            score , diff =
compare_ssim(warped_image[cropfrom:c
ropto,cropfrom:cropto],org_image[cro
```

```
run()
```

کد قسمت 7.2

```
from skimage.io import imread
from skimage.color import rgb2gray
from scipy import signal as sig
import numpy as np
import scipy.ndimage as ndi
import cv2
import matplotlib.pyplot as plt


def gradient_x(imggray):
    ##Sobel operator kernels.
    kernel_x = np.array([[-1, 0,
1],[-2, 0, 2],[-1, 0, 1]])
    return sig.convolve2d(imggray,
kernel_x, mode='same')
def gradient_y(imggray):
    kernel_y = np.array([[1, 2, 1],
[0, 0, 0], [-1, -2, -1]])
    return sig.convolve2d(imggray,
kernel_y, mode='same')


def run():
    data = []
    scales = [1,2,3,4]
    for scale in scales:
        img =
cv2.imread('../image/Building.jpg')
        img =
cv2.resize(img,(int(img.shape[1]/sca
le),int(img.shape[0]/scale)))
        imggray = rgb2gray(img)
        I_x = gradient_x(imggray)
        I_y = gradient_y(imggray)

        Ixx =
ndi.gaussian_filter(I_x**2, sigma=1)
        Ixy =
ndi.gaussian_filter(I_y*I_x,
sigma=1)
        Iyy =
ndi.gaussian_filter(I_y**2, sigma=1)


        k = 0.05

        # determinant
        detA = Ixx * Iyy - Ixy ** 2
        # trace
        traceA = Ixx + Iyy

        harris_response = detA - k *
traceA ** 2

        img_copy_for_corners =
np.copy(img)
        img_copy_for_edges =
np.copy(img)
```

```
pfrom:cropto,cropfrom:cropto],full=T
rue)
            mmse          =
mean_squared_error(warped_image[crop
from:cropto,cropfrom:cropto],org_ima
ge[cropfrom:cropto,cropfrom:cropto])
            mp            =
len(good_matches)

data.append([img_name,score,mmse,mp]
)
        else:

cv2.imwrite('../image/hw7.1/Attack-
warped{}.jpg'.format(img_name),
warped_image)

cv2.imwrite('../image/hw7.1/Attack-
diff{}.jpg'.format(img_name),org_ima
ge - warped_image)
            score , diff =
compare_ssim(warped_image,org_image,
full=True)
            mmse          =
mean_squared_error(warped_image,org_
image)
            mp            =
len(good_matches)

data.append([img_name,score,mmse,mp]
)
    npArray_data =
(np.array(data))[:,1:].astype(np.flo
at)
    mean =
["mean",statistics.mean(npArray_data
[:,0]),statistics.mean(npArray_data[
:,1]),statistics.mean(npArray_data[:
,2])]
    std =
["stdev",statistics.stdev(npArray_da
ta[:,0]),statistics.stdev(npArray_da
ta[:,1]),statistics.stdev(npArray_da
ta[:,2])]
    data.append(mean)
    data.append(std)

    #display mmse table
    types = [" ","ssim","mmse","mp"]
    col_labels = tuple(types)
    fig, ax = plt.subplots(dpi=300,
figsize=(5,5))
    ax.axis('off')
    ax.table(cellText=data,
colLabels=col_labels, loc='center')

fig.savefig('../image/hw7.1/resTable
.png')
    #
    # img3 =
cv2.drawMatchesKnn(img1,kp1,img2,kp2
,good_matches,None,flags=cv2.DrawMat
chesFlags_NOT_DRAW_SINGLE_POINTS)
    # plt.imshow(img3),plt.show()
```

```
        threshold_corners = 0.1
        threshold_edges = -0.1
        corner_counter = 0
        for rowindex, response in
enumerate(harris_response):
            for colindex, r in
enumerate(response):
                if r >
threshold_corners:
                    # this is a
corner

img_copy_for_corners[rowindex,
colindex] = [255,0,0]
                    corner_counter
+= 1
                elif r <
threshold_edges:
                    # this is an
edge

img_copy_for_edges[rowindex,
colindex] = [0,255,0]



cv2.imwrite('../image/hw7.2/hariss-
corner-
scale{}.png'.format(scale),img_copy_
for_corners)

cv2.imwrite('../image/hw7.2/hariss-
edge-
scale{}.png'.format(scale),img_copy_
for_edges)



        #this part is for
non_maximal_supperssion
        img_copy_for_corners =
np.copy(img)
        img_copy_for_edges =
np.copy(img)

        threshold_corners = 0.1
        threshold_edges = -0.1
        size_tilex = 3
        size_tiley = 3
        half_size_tilex =
size_tilex//2
        half_size_tiley =
size_tiley//2
        corner_counter_non_max = 0
        for rowindex in range(
half_size_tiley  ,
((harris_response.shape[0]) -
half_size_tiley)    ):
            for colindex in range(
half_size_tilex  ,
```

```
(harris_response.shape[1]) -
half_size_tilex    ):
                tile_corner =
harris_response[  rowindex-
half_size_tiley
:rowindex+1+half_size_tiley ,
colindex-
half_size_tilex:colindex+1+half_size
_tilex]
                maxr =
np.max(tile_corner)
                r =
harris_response[rowindex,colindex]
                if r >
threshold_corners and r >= maxr:
                    # this is a
corner

img_copy_for_corners[rowindex,
colindex] = [0,0,255]

corner_counter_non_max += 1

cv2.imwrite('../image/hw7.2/hariss-
corner_non_max-
scale{}.png'.format(scale),img_copy_
for_corners)



data.append([scale,corner_counter,co
rner_counter_non_max])

    #display mmse table
    types = [" ","harris","non
maximal suppersion"]
    col_labels = tuple(types)
    fig, ax = plt.subplots(dpi=300,
figsize=(5,5))
    ax.axis('off')
    ax.table(cellText=data,
colLabels=col_labels, loc='center')

fig.savefig('../image/hw7.2/resTable
.png')

run()
print("finished")
```

مراجع

کتاب گنزالس رافائل