ویژگی‌ها

رضا عباس‌زاده

اطلاعات گزارش

تاریخ: ۱۴۰۰/۰۴/۱۰

واژگان کلیدی:
نقاط ویژگی
توصیف‌کننده
تخمین هندسی
گوشه یاب هریس
Sift
Surf

چکیده

در این تمرین نقاط ویژگی تصویر و نحوه شناسایی آن‌ها توضیح داده شده است. سپس تخمین هندسی برای انواع حالات مختلف یک تصویر بررسی می‌شود. در نهایت با استفاده از الگوریتم گوشه‌یاب هریس، گوشه‌های موجود در یک تصویر را شناسایی می‌کنیم.

۱-مقدمه

ویژگی، یک مشخصه بصری از تصویر است که می‌تواند از رنگ، بافت، شکل یا لبه‌های تصویر استخراج بشود. در واقع، استخراج ویژگی، فرایند تبدیل مقدارهای خام پیکسل‌های یک تصویر به اطلاعات مفید و معنادار است. در گذشته، طراحی این فرایند توسط متخصصان این حوزه انجام میشده- است و باید ویژگی‌ها رو به صورت دستی تعریف و استخراج می‌کردیم ولی امروزه با کمک شبکه‌های عصبی و یادگیری عمیق، این عمل به صورت خودکار در لایه‌های کانولوشنی شبکه‌های عصبی انجام میشه. از ویژگی‌ها در وظایفی مثل بخش‌بندی (segmentation) یا تشخیص اشیاء (object recognition) استفاده میشه.

ویژگی‌های استخراج شده از تصاویر، در قالب مقادیر عددی نشان داده می‌شوند و معمولا نسبت به تصویر اصلی، ابعاد بسیار کمتری دارند. معیارهای انتخاب یک ویژگی:

- ✓ **کارایی**: یک ویژگی باید علاوه بر اینکه قدرت تمایز بین اشیاء رو به ما بدهد، خیلی سریع در تصویر شناسایی و محاسبه بشود.
- ✓ **مقیاس‌پذیری**: عملکرد سیستم تحت تاثیر سایز پایگاه داده قرار نگیرد. مخصوصا که معمولا ما با پایگاه داده‌های چندین بعدی و بسیار بزرگ سروکار داریم.
- ✓ **پایداری**: نسبت به تغییر شرایط تصویر مانند تغییر زاویه نور یا زاویه دید مقاوم باشد و سعی کنیم

تغییر در شرایط ویژگی، تا حد امکان، روی تصمیم‌گیری مدل اثر نگذارد.

به صورت کلی، ویژگی‌های استخراج شده از تصاویر به دو دسته تقسیم می‌شوند: سراسری و محلی.

۱-۱-ویژگی های سراسری

این ویژگی‌ها را میتونیم از تصویر خام، در گام اول پردازش تصویر استخراج کنیم. آن‌ها معمولا نسبت به مواردی همچون تغییر زاویه نور پایدار نیستند. در بازیابی تصویر از این ویژگی‌ها برای «شناسایی اشیاء» (بررسی حضور یا عدم حضور یک شی خاص در تصویر) استفاده می‌شود. البته صرفا استفاده از این ویژگی‌ها برای دریافت محتوا و معنای یک تصویر کافی نیست.

۱-۲-ویژگی های محلی

بعد از استخراج ویژگی‌های سراسری و شناسایی قسمتی از تصویر که احتمال حضور یک شی در اون وجود داره، تشکیل کادرهای Region of Interest= RoI، با تقسیم آن بخش‌ها، به بلوک‌هایی کوچکتر، به بررسی جزئی‌تر ویژگی‌ها در هر یک از بلوک‌ها می‌پردازیم تا ماهیت آن شی (برچسب اون) تشخیص داده بشه. استخراج این ویژگی‌ها، برای توصیف محتوای تصویر ضروری است.

۲-شرح تکنیکال

۱-۷ Estimate geometry

این روش در سه مرحله انجام می شود: شناسایی نقاط ویژگی و توصیف آن‌ها، تطبیق ویژگی ها، پیدا کردن تابع تبدیل.
به طور خلاصه ، ما نقاط ویژگی 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 را طوری تغییر می دهیم که نقاط ویژگی دو تصویر روی هم قرار بگیرند.

۱-۱-۷-پیدا کردن نقاط ویژگی و توصیف آن‌ها:

نقاط ویژگی یک تصویر شامل گوشه‌ها، لبه‌ها و... مهم و قابل تمایز می‌باشد. بردار ویژگی که به عنوان توصیف کننده آن نقطه ویژگی است، شامل ویژگی‌های مهم نقاط ویژگی است که باید در برابر انواع تغییرات(بزرگنمایی، تغییرات نور و...) مقاوم باشد.
الگوریتم‌های مختلفی برای شناسایی و توصیف نقاط ویژگی وجود دارند:
الگوریتم sift یکی از بهترین این الگوریتم‌ها است اما به صورت رایگان در اختیار همه قرار نگرفته است. این الگوریتم در برابر بزرگنمایی، تغییرات روشنایی، جهت گیری ها مقامت بالایی دارد.
الکوریتم surf از sift الهام گرفته شده است اما به دلیل استفاده از تقریب با استفاده از تصویر انتگرالی و فیلتر جعبه‌ای و همچنین قابلیت موازی‌سازی بسیار سریعتر است. با این حال مقداری کاهش مقاومت نسبت به نورپردازی و چرخش تصاویر دارد. این الگوریتم رایگان قابل استفاده است.

۲-۱-۷-تطبیق نقاط ویژگی:

پس از مشخص شدن نقاط 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 استفاده از knnMatch است. این روش فاصله بین هر جفت توصیف کننده(descriptor) را اندازه گیری می کند و برای هر نقطه کلیدی k تا بهترین خود را با حداقل فاصله بازمی گرداند. سپس یک ترشولد بر روی فاصله بین ویژگی ها گرفته میشود و جفت‌هایی با کمترین فاصله را انتخاب میکنیم.

۳-۱-۷-پیدا کردن ماتریس تبدیل:

پس از تطبیق حداقل چهار جفت نقطه ویژگی ، می‌توانیم 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

تبدیلات هندسی هستند که دارای ۸ پارامتر هستند و توسط یک ماتریس مربعی ۳ در ۳ نشا 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 تبدیل شده ، ما ماتریس هموگرافی را محاسبه می کنیم و آن را بر روی تصویر سنجیده اعمال می کنیم.

Corner detection -۷-۲

الگوریتم گوشه‌یاب هریس تنها قادر به تشخیص گوشه‌های یک تصویر می‌باشد و امکان توصیف و تطبیق گوشه‌های دو تصویر را ندارد.

این الگوریتم مشابه الگوریتم های بالا انرژی حاصل از جابه‌جایی پنجره ای که روی حرکت حرکت می‌دهیم را در نظر می گیرد.

اگر تغییرات انرژی در دو جهت متفاوت مقدار بالایی داشته باشد ینی پنجره روی یک گوشه قرار دارد.

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

Window function — Shifted intensity — Intensity

با ساده‌سازی عبارت بالا به فرمول‌های زیر می‌رسیم:

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

$$M=\sum_{x,y} w(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

در رابطه بالا درایه‌های ماتریس حاصلضرب مشتقات در دو دو جهت x و y می‌باشد.

با داشتن این ماتریس‌ها به محاسبه مقدار R می‌پردازیم:

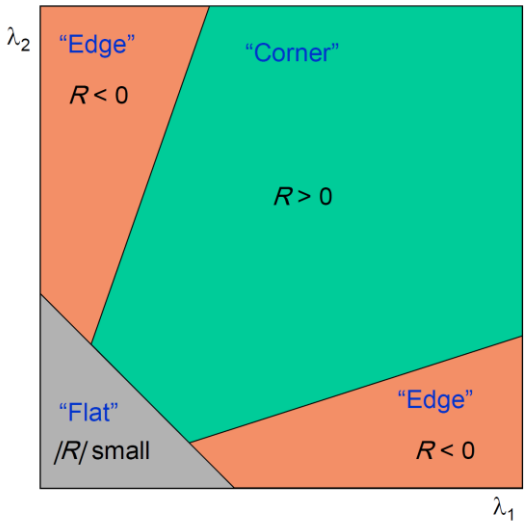
$$R=\det M-\alpha\left(\operatorname{trace} M\right)^2$$

$$\det M=\lambda_1\lambda_2$$
$$\operatorname{trace} M=\lambda_1+\lambda_2$$

در نتیجه داریم:

$$R=\det(M)-\alpha\operatorname{trace}(M)^2=\lambda_1\lambda_2-\alpha(\lambda_1+\lambda_2)^2$$

و با توجه به تصویر زیر گوشه‌ها و لبه‌ها را شناسایی می کنیم:



سپس با قرار دادن ترشولد برای مقدار R نقاط اضافه را حذف می کنیم.

Non-maximum suppression

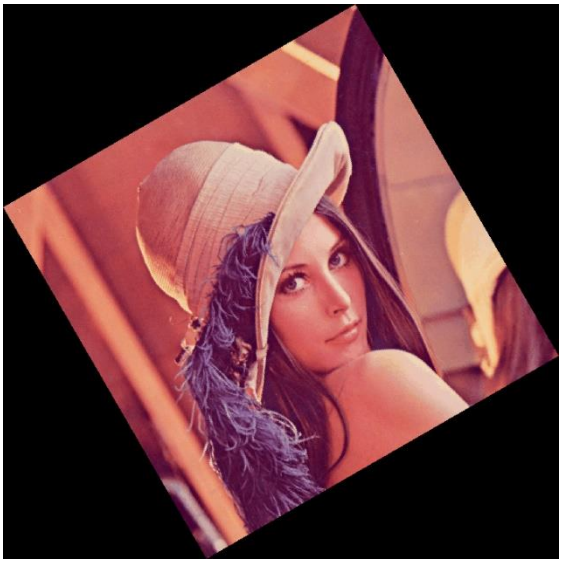
در این روش گوشه‌های به دست آمده در روش هریس را جدا می‌نیم. سپس روی این تصویر پنجره ۳ در ۳ را کانوالو می کنیم. ماکزیمم محلی داخل این پنجره که مقداری بیشتر از مقدار ترشولد دارد را نگه می‌داریم و بقیه نقاط چنانچه گوشه بودند، از لیست گوشه‌ها حذف می‌شوند. به این ترتیب از ترکیب شدن جندین گوشه و زیاد شدن تعداد آن‌ها جلوگیری کرده‌ایم.

۳-شرح نتایج

۷-۱ تصویر اصلی:

تصویر مرجع:

تصویر اول:



جدول نتایج و معیارهای ssim, mse, mp:

	ssim	mse	mp
Original.bmp	0.9765491829004282	7.0281524658203125	381
1.bmp	0.9182357847700755	47.71928405761719	146
2.bmp	0.8439816875358995	93.29692840576172	169
3.bmp	0.906529681253241	159.88822518725334	158
4.bmp	0.7672057910379101	413.92201232910156	81
mean	0.8825004254995109	144.37092048911083	187.0
std	0.07980622737743372	161.0215088720612	113.70795926407263

در سطر اول این جدول ابتدا کل نقاط ویژگی را پیدا کرده‌ایم و تطبیق داده ایم. همانطور که می‌بینید ۳۸۱ نقطه ویژگی به دست آمده‌است. برای انتخاب ترشولد شباهت، اعداد مختلف را امتحان کردم و با ترشولد ۰.۷۴ کمترین میزان خطا و بیشترین نقطه ویژگی به دست آمد و برای بقیه عکس‌های تغییر یافته هم از همین مقدار ترشولد استفاده کرده‌ام.

تصویر بازسازی شده اول:

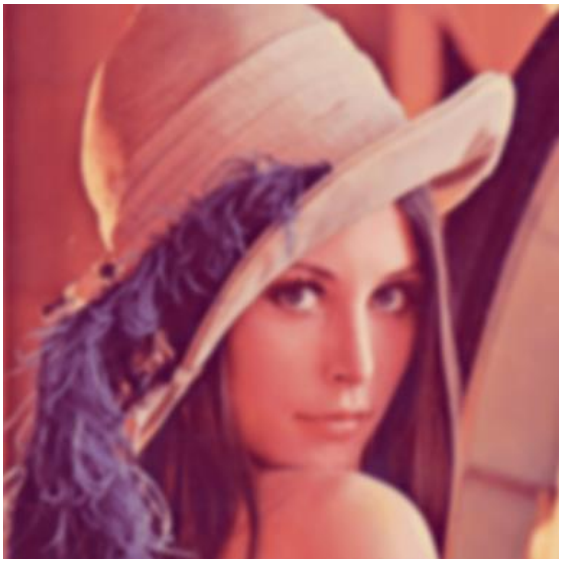
این تصویر همزمان چرخش و متعادل‌سازی هستوگرام انجام شده است. با توجه به اینکه الگوریتم surf نسبت به تغییرات نورپردازی مقاومت کمی دارد، می‌بینیم که تعداد نقاط ویژگی به دست آمده به ۱۴۶ عدد کاهش یافته است. با این‌حال نتیجه قابل‌قبول است و تاثیرات چرخش تصویر به طور کامل از بین رفته است.

تصویر دوم:

تصویر بازسازی شده دوم:

همانطور که در جدول دیدید، تعداد نقاط ویژگی در این مورد ۱۸۳ عدد می‌باشد که نسبت به تصویر قبل بیشتر شده است. بنابراین نتیجه میگیریم که تبدیل sharp و کشیده شدن تصویر کمتر به شناسایی نقاط ویژگی آسیب می‌زنند. با این وجود کمی خطا به تصویر بازسازی شده اضافه شده است اما خیلی شدید نبوده است.

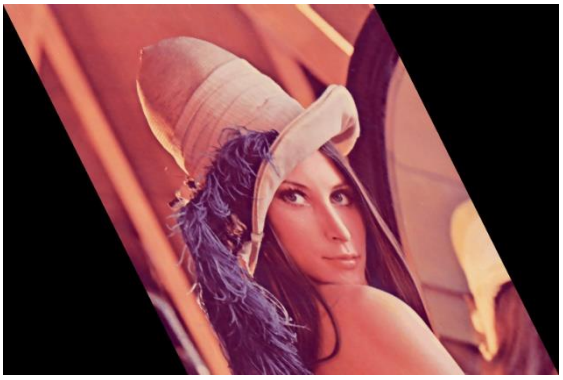
تصویر سوم:



تصویر بازسازی‌شده سوم:

می‌بینیم که این الگوریتم نسبت به تصاویر تار شده مقاوت خوبی نشان می‌دهد و با توجه به مقدار خوب ssim ساختار کلی تصویر به خوبی بازسازی شده است. اما خطای به دست آمده افزایش چشمگیری داشته است و تصویر مقداری از بین رفته است.

تصویر چهارم:



تصویر بازسازی‌شده چهارم:

با توجه به اطلاعات جدول در این مورد تنها ۹۱ نقطه ویژگی به دست آمده است که کاهش زیادی نسبت به حالات دیگر داشته است. این نشان‌دهنده مقاومت کم surf نسبت به کشیدگی در دو جهت است. از طرفی مقدار خطا بسیار افزایش یافته است و مقدار ssim تا ۷۴ درصد افت داشته است.

corner detection ۷-۲

گوشه‌های به دست آمده از روش هریس:

بزرگنمایی بخشی از تصویر:

می‌بینیم که گوشه‌های زیادی در تصویر هستند که شناسایی نشده‌اند. بنابراین ترشولد را کمتر کردم:

می‌بینیم که گوشه‌های از دست رفته کمتر شده اند اما تعداد گوشه‌ها بسیار بیشتر شده اند و پردازش آن‌ها هزینه بیشتری خواهد داشت. مثلا در برگ‌های درختان تعداد زیادی گوشه شناسایی شده است.

نقاط باقی‌مانده بعد از اعمال non max suppression:

می‌بینید که در این روش گوشه‌های شناسایی شده از یکدیگر جدا شده‌اند و از هر دسته نقطه‌ای که در روش قبل گوشه شناسایی شده اند، با استفاده از یک فیلتر ۳ در ۳، یک نقطه انتخاب شده است.

در روش اول ۱۲۲۴۰۴ عدد گوشه در تصویر شناسایی شد در حالی که بعد از اعمال non max suppression این تعداد به ۷۵۸۹ کاهش یافت.

٤-پیوست

کد مربوط به سوال اول

```
img_names = ['Original.bmp', '1.bmp', '2.bmp', '3.bmp', '4.bmp']
original = cv2.imread('Original.bmp', cv2.IMREAD_GRAYSCALE)
ref = cv2.imread('Reference.bmp', cv2.IMREAD_GRAYSCALE)
data = []
for img_name in img_names:
    img_attack1 = cv2.imread('Attack 1/{}'.format(img_name),
cv2.IMREAD_GRAYSCALE)
    img_attack2 = cv2.imread('Attack 2/{}'.format(img_name),
cv2.IMREAD_GRAYSCALE)
    surf = cv2.xfeatures2d.SURF_create(400)
    kp1, des1 = surf.detectAndCompute(ref, None)
    kp2, des2 = surf.detectAndCompute(img_attack1, None)
    bf = cv2.BFMatcher()
```

```
    matches = bf.knnMatch(des1, des2, k=2)
    good_matches = []
    for m1, m2 in matches:
        if m1.distance < 0.74 * m2.distance:
            good_matches.append([m1])
    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])
    H, status = cv2.findHomography(sensed_matched_kpts, ref_matched_kpts,
cv2.RANSAC, 5.0)
    warped_image = cv2.warpPerspective(img_attack2, H, (ref.shape[1],
ref.shape[0]))
    if img_name == '3.bmp':
        crop_start = 63
        crop_end = warped_image.shape[0] - crop_start

        cv2.imwrite('7-1/Attack-warped{}.jpg'.format(img_name),
                    warped_image[crop_start:crop_end, crop_start:crop_end])
        cv2.imwrite('7-1/Attack-diff{}.jpg'.format(img_name),
                    original[crop_start:crop_end, crop_start:crop_end] -
warped_image[crop_start:crop_end, crop_start:crop_end])
        score, diff = compare_ssim(warped_image[crop_start:crop_end,
crop_start:crop_end],
                                   original[crop_start: crop_end,
crop_start: crop_end], full=True)
        mse = mean_squared_error(
            warped_image[crop_start:crop_end, crop_start:crop_end],
            original[crop_start:crop_end, crop_start:crop_end])
        mp = len(good_matches)
        data.append([img_name, score, mse, mp])
    else:
        cv2.imwrite('7-1/Attack-warped{}.jpg'.format(img_name),
warped_image)
        cv2.imwrite('7-1/Attack-diff{}.jpg'.format(img_name), original -
warped_image)
        score, diff = compare_ssim(warped_image, original, full=True)
        mse = mean_squared_error(warped_image, original)
        mp = len(good_matches)
        data.append([img_name, score, mse, mp])


npArray_data = (np.array(data))[:, 1:].astype(float)
mean = ["mean", statistics.mean(npArray_data[:, 0]),
        statistics.mean(npArray_data[:, 1]),
        statistics.mean(npArray_data[:, 2])]
std = ["std", statistics.stdev(npArray_data[:, 0]),
       statistics.stdev(npArray_data[:, 1]),
       statistics.stdev(npArray_data[:, 2])]
data.append(mean)
data.append(std)
types = [" ", "ssim", "mse", "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('7-1/res.png')
```

کد مربوط به سوال دوم

```
img = cv2.imread('Building.jpg')
img = cv2.resize(img, (int(img.shape[1]), int(img.shape[0])))
gray_img = rgb2gray(img)
I_x = gradient_x(gray_img)
I_y = gradient_y(gray_img)
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
detA = Ixx * Iyy - Ixy ** 2
traceA = Ixx + Iyy
harris_response = detA - k * traceA ** 2
img_with_corners = np.copy(img)

threshold_corners = 0.2
corners_count = 0
for rowindex, response in enumerate(harris_response):
    for col, r in enumerate(response):
        if r > threshold_corners:
            img_with_corners[rowindex, col] = [30, 30, 200]
            corners_count += 1

print("harris corner counts:", str(corners_count))
cv2.imwrite('7-2/harris-corner.png', img_with_corners)

img_with_corners = np.copy(img)
img_copy_for_edges = np.copy(img)
window_w = 3
window_h = 3
window_w_half = window_w // 2
window_h_half = window_h // 2
corners_count = 0
for rowindex in range(window_h_half, ((harris_response.shape[0]) -
window_h_half)):
    for col in range(window_w_half, (harris_response.shape[1]) -
window_w_half):
        tile_corner = harris_response[rowindex - window_h_half:rowindex + 1
+ window_h_half,
                      col - window_w_half:col + 1 + window_w_half]
        max_in_row = np.max(tile_corner)
        r = harris_response[rowindex, col]
        if r > threshold_corners and r >= max_in_row:
            img_with_corners[rowindex, col] = [30, 30, 200]
            corners_count += 1
print("non suppression corner counts:", str(corners_count))
cv2.imwrite('7-2/non_max.png', img_with_corners)
```

نحوه محاسبه گرادیان در جهت‌های x و y:

```
def gradient_x(img):
    kernel_x = np.array([
        [-1, 0, 1],
        [-2, 0, 2],
        [-1, 0, 1]
    ])
```

```
    return sig.convolve2d(img, kernel_x, mode='same')


def gradient_y(img):
    kernel_y = np.array([
        [1, 2, 1],
        [0, 0, 0],
        [-1, -2, -1]
    ])
    return sig.convolve2d(img, kernel_y, mode='same')
```