Features

علیرضا لرستانی

اطلاعات گزارش

تاریخ: **11/05**/1399

واژگان کلیدی:
Feature points
Harris corner detector
Surf
Sift

چکیده

در این تمرین با مبحث نقاط ویژگی آشنا شده و یاد میگریم چگونه آن ها را مشخص کنیم. سپس با برخی از کاربرد های آن ها در متد های پردازش تصویر آشنا میشویم و یکی از الگوریتم های مهم در امر تشخیص نقاط ویژگی، یعنی الگوریتم هریس را پیاده سازی میکنیم.

۱-مقدمه

برای انجام عملیات های مختلف بر روی دو تصویر جداگانه اعم از تطبیق، مقایسه، بازسازی و بسیاری از عملیات های دیگر بر روی تصاویر، نیاز به نقاط خاصی داریم که بتوانند به خوبی تصویر را برای ما توصیف کنند.

این نقاط، نقاط ویژگی یا feature point نامیده میشوند که باید دارای ویژگی های خاصی باشند تا بتوانند حداکثر دقت در توصیف تصویر را به ارمغان بیاورند.

الگوریتم های متعددی برای محاسبه این نقاط و همچنین توصیف آن ها وجود دارد که هر یک داری برتری ها و همچنین نقاط ضعفی هستند.

الگوریتم گوشه یاب هریس که در ادامه با آن بیشتر آشنا میشویم یکی از این الگوریتم هاست که به امر detection یا تشخیص نقاط ویژگی می پردازد.

نوشتار حاضر، به بررسی و پیاده سازی (با زبان برنامه نویسی متلب) تعدادی عملیات پردازش تصویر در باب نقاط ویژگی میپردازد.

۲-شرح تکنیکال

تا کنون توضیح مختصری درباره ی نقاط ویژگی و اینکه چه کاربردی برای ما دارند داده شد.

- **اما یک نقطه ویژگی خوب چه نقطه ای است؟**

مطابق تعریف و کاربرد بیان شده نقطه ویژگی خوب باید دارای ویژگی های زیر باشد:

1- منحصر به فرد باشد.

2- تکرار پذیر باشد.

* پست الکترونیک نویسنده مسئول:
alirezalorestani2010@gmail.com

1. دانشجو، دانشکده مهندسی، دانشگاه فردوسی مشهد

3- به texture پشت وابسته نباشد.

4- و در نهایت، جزو بافت تصویر باشد.

ساخت تصاویر پانوراما، جستجوی اشیا در تصویر و تطابق در تصویر از اصلی ترین و ساده ترین کاربرد های نقاط ویژگی است.

به دست آوردن و کار کردن با نقاط ویژگی سه مرحله ی اصلی دارد:

1 - تشخیص یا detection

2 - توصیف یا description

3 - تطابق یا matching

در گذشته با لبه ها در تصویر آشنا شدیم و در فرایند های متعددی از آن ها بهره بردیم، حال بیایید بررسی کنیم و ببینیم آیا لبه ها میتوانند توصیفگر خوبی با توجه به ویژگی های بیان شده باشند:

در تصویر بالا سه نقطه تحت عنوان لبه را با کادر قرمز مشخص کرده ایم.

به طور مثال به دو نقطه ی سمت راستی دقت کنید:

تفاوتی بین این دو نقطه میبینید؟

پاسخ منفی است. حال بیایید نتیجه گیری کلی ای در این باره انجام دهیم.

اگر کادر مشخص شده را در جهت لبه ها جابجا کنیم تغییر محسوسی را احساس نخواهیم کرد.

در نتیجه اصل منحصر به فرد بودن نقض شده چرا که تمامی این نقاط منحصر به فرد یکسان تشخیص داده میشوند.

حال به نقطه سبز رنگ مشخص شده در تصویر دقت کنید.

این نقطه نقطه ای است که ما تحت عنوان گوشه از آن یاد میبریم.

این نقاط بسیار کارامد تر و بهتر از لبه ها هستد میتوانند کاندید خوبی برای انتخاب به عنوان نقطه ی ویژگی هستند.

چرا که مشکلی که در لبه ها وجود داشت اینجا تماما برطرف شده.

کادر سبز رنگ را در هر جهتی جا به جا کنیم تغییرات محسوسی را مشاهده خواهیم کرد.

و دیگر احتمال overlap بین نقاط همسایه برطرف میشود.

گرچه که باز هم نمیتوان گفت هر نقطه ی گوشه ای بهترین نقطه ی ویژگی ممکن است. مثلا در همین تصویر کادر قرمز رنگ در بالای تصویر گوشه ای است که در تصویر به دفعات تکرار شده و میتواند ما را دچار خطا در تطابق نقاط کند.

در ادامه به معرفی الگوریتم گوشه یاب هریس میپردازیم.

این الگوریتم تنها در امر detection کاربرد دارد و راه حلی برای description و matching ارائه نمیدهد.

الگوریتم گوشه یاب هریس

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

فرمول بالا دقیقا همان توصیف ریاضی ای از آنچه پیش تر توضیح دادیم است.

انرژی ای به هر نقطه (x, y) نسبت میدهد که مشخص کننده میزان تغییرات پنجره ی (x, y)w بر روی تصویر (x, y)I به اندازه (u, v) است.

با بسط این رابطه از طریق سری تیلور به رابطه ی زیر میرسیم:

$$E(u,v) \approx [u \;\; v]\; M \begin{bmatrix} u \\ v \end{bmatrix}$$

که در آن M از رابطه ی زیر به دست می آید:

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

درایه های ماتریس بالا مشتقات تصویر در جهت های مختلف و حاصل ضرب های آن هاست.

در نهایت با داشتن ماتریس M، به محاسبه ی مقدار R میپردازیم:

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

$$\det M=\lambda_1\lambda_2$$

$$trace\,M=\lambda_1+\lambda_2$$

با آستانه گذاری مناسب بر روی این مقدار به دست آمده، نقاطی که مقدار R محاسبه شده برای آن ها از مقداری مشخص بیشتر باشد، گوشه هستند.

هریس نسبت به translate، photometrics و rotaion مقاوم است اما اگر تصویر ورودی scale داده شود دیگر قادر به تشخیص نقاط مشترک نخواهد بود.

در سال ۲۰۰۴ David G.Lowe الگوریتمی ارائه کرد که بتواند مشکلات هریس از جمله ناتوانی در تشخیص نقاط scale شده و همچنین describtion و matching را پوشش دهد.

این الگوریتم SIFT یا Scaling Invariant Feature Transform نام دارد.

خلاصه ای از مراحل این الگوریتم را در زیر مرور میکنیم:

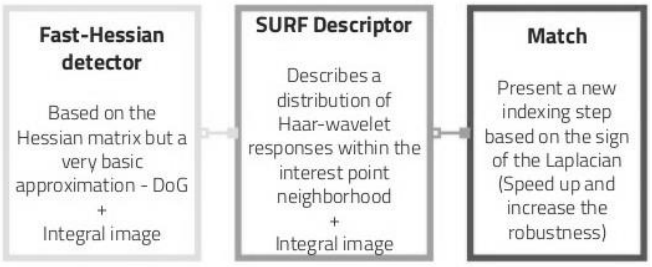
1- Find Scale-Space Extrema
2- Keypoint Localization & Filtering (Improve keypoints and throw out bad ones)
3- Orientation Assignment (Remove effects of rotation and scale)
4- Create descriptor(Using histograms of orientations)

۲ فاز اول این الگوریتم معادل اند با همان عمل detection و فاز ۳و ۴ معادل اند با description.

در نهایت با انجام مراحل بالا میتوانیم به جای ذخیره تصاویر اصلی، تنها نقاط ویژگی آن ها را ذخیره کنیم که فضای بسیار کمتری اشغال میکنند و همچنین میتوانند تصویر را به خوبی توصیف کنند.

از مشکلات این الگوریتم میتوان به سرعت پایین آن اشاره کرد. عاملی که باعث شد در سال ۲۰۰۶ الگوریتم SURF یا Speeded Up Robust Features ارائه شود.

در تصویر زیر خلاصه ای از مراحل این الگوریتم را در زیر مرور میکنیم:



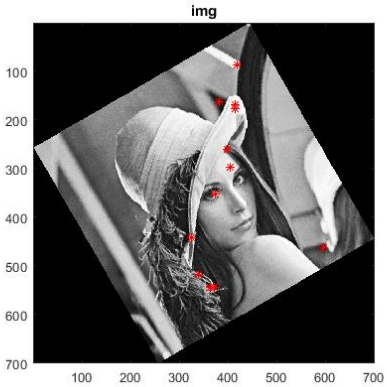
۳-شرح نتایج و نتیجه گیری

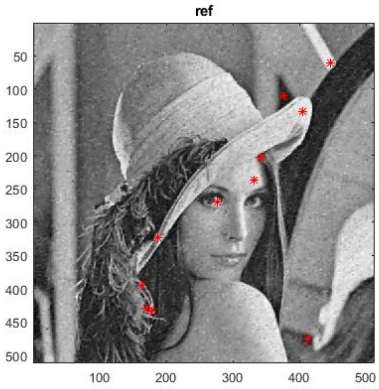
۷.۱.۱

ابتدا تصویر refrence را graysacle کرده :

سپس تصویر اصلی که تصاویر پس از بازسازی با آن مقایسه خواهند شد را نیز مشاهده میکنیم:

ابتدا تصویر اول را انتخاب میکنیم و پس از به دست آوردن نقاط ویژگی به بازسازی تصویر میپردازیم:

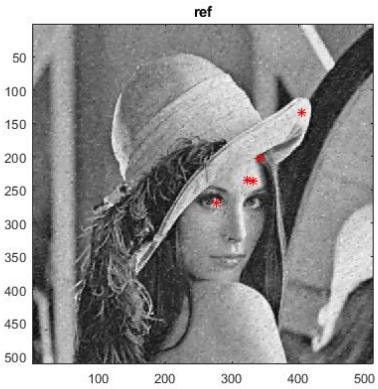




تصویر بازسازی شده:

SSIM	MSE	MP
0.8199	81.8375	12

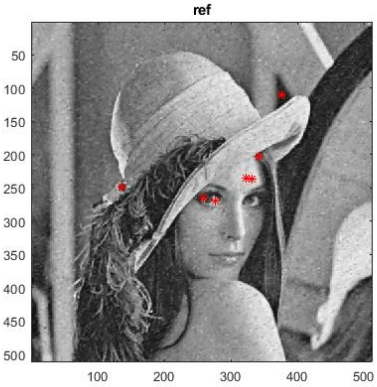
سپس به سراغ تصویر دوم میرویم:

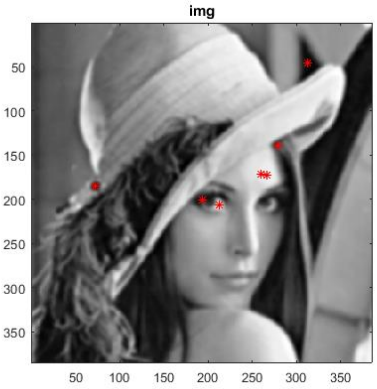


تصویر بازسازی شده:

SSIM	MSE	MP
0.53	592.6	6

سپس به سراغ تصویر سوم میرویم:

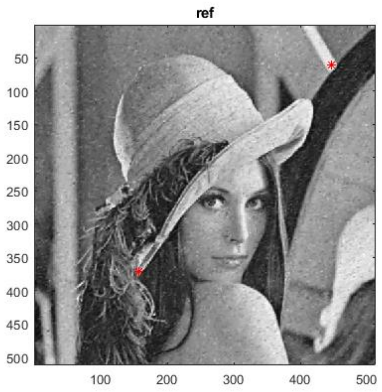


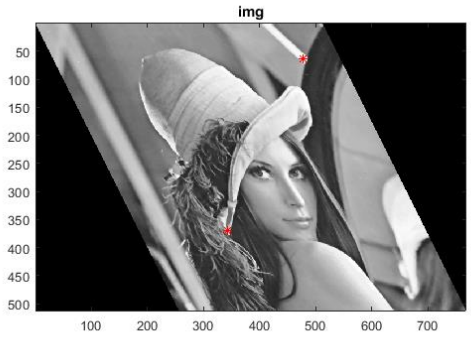


تصویر بازسازی شده:

SSIM	MSE	MP
0.59	390.4	7

و در نهایت تصویر چهارم:





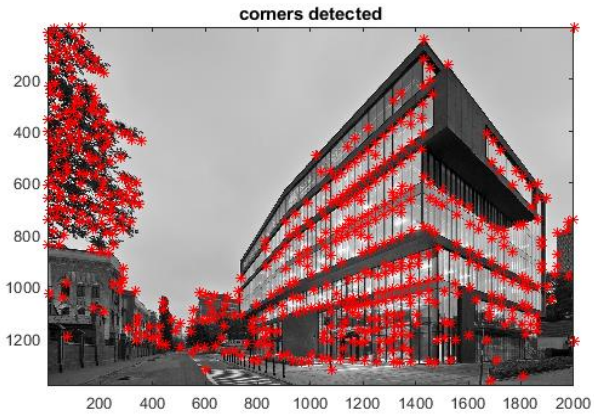
. در نهایت تصویر نتیجه:

SSIM	MSE	MP
0.3820	2911.7	2

و در نهایت مقادیر آماری بر روی داده های به دست آمده به صورت زیر است:

	SSIM	MSE	MP
Mean	0.58	994.13	6.75
Std	0.15	1121.94	3.56

۷.۲.۱



۴-پیوست

- ۷.۱.۱

```
feature_img = detectSURFFeatures(img,'MetricThreshold',mt);
feature_ref = detectSURFFeatures(ref,'MetricThreshold',mt);


[f1,vpts1] = extractFeatures(img,feature_img);
[f2,vpts2] = extractFeatures(ref,feature_ref);

indexPairs = matchFeatures(f1,f2);
matchedPoints1 = vpts1(indexPairs(:,1));
matchedPoints2 = vpts2(indexPairs(:,2));


x1 = matchedPoints1.Location(:,1);
y1 = matchedPoints1.Location(:,2);
x2 = matchedPoints2.Location(:,1);
y2 = matchedPoints2.Location(:,2);
z1 = [x1 y1];
z2 = [x2 y2];

T = fitgeotrans(z1,z2,'nonreflectivesimilarity');
```

- ۷.۲.۱

```
dx = [-1 0 1;-1 0 1;-1 0 1]/6;
dy = [-1 -1 -1;0 0 0 ; 1 1 1]/6;
sigma = 0.5;
g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);

Ix = filtering(img,dx);
Iy = filtering(img,dy);
Ix2 = conv2(Ix.^2, g);
Iy2 = conv2(Iy.^2, g);
Ixy = conv2(Ix.*Iy, g);

k = 0.04;
R11 = (Ix2.*Iy2 - Ixy.^2) - k*(Ix2 + Iy2).^2;
thresh = 10;
cond = R11>thresh;
R = R11.*cond;
R = norm(R);
imshow(R);
[r,c] = nms(R,20,thresh);
figure, imagesc(img), axis image, colormap(gray), hold on
    plot(c,r,'r*'), title('corners detected');
```

```
function [r,c] = nms(cim,radius,thresh)
    sze = 2*radius+1;
    mx = ordfilt2(cim,sze^2,ones(sze));
    cim = (cim==mx)&(cim>thresh);

    [r,c] = find(cim);

end
function output = norm(img)
    output = mat2gray(img);
    Max = max(max(output));
    Min = min(min(output));
    output = (255/(Max-Min))*output;

end
```

```
function output=filtering(img,mask)
    [R,C] = size(img);
    [x,y] = size(mask);
    img = padarray(img,[1,1]);
    output = zeros(R,C,'double');
    for i=1:R
        for j=1:C
            part = img(i:i+x-1,j:j+y-1);
            mult = part.*mask;
            out = sum(mult,'all');
            output(i,j) = out;
        end
    end
end
```