گزارش تمرین Features

مریم واقعی

اطلاعات گزارش

تاریخ:
۱۸ بهمن ۱۴۰۱

واژگان کلیدی:
نقطه ویژگی
گوشه یاب هریس
SIRF
گوشه یابی
استخراج ویژگی

چکیده

نقاط ویژگی از پایه ترین ابزارها در کاربردهای بینایی مشاین هستند. می‌توان به جای ذخیره تصاویر، بردار ویژگی آن ها را ذخیره نمود و از این بردار ها برای تشخیص ماهیت یک عکس استفاده کرد.

یکی از نقاط ویژگی مهم در تصاویر، گوشه ها هستند. الگوریتم گوشه‌یاب هریس، با استفاده از یک پنجره که تصویر را پیمایش می‌کند، مقدار گرادیان آن را محاسبه می‌کند و سپس با استفاده از بسط تیلور، تخمینی از آن را به دست می‌آورد. از مقادیر ویژه ماتریس به دست آمده از بارت تخمینی، برای شناسایی گوشه ها در تصویر استفاده می‌شود.

یکی از الگوریتم های استخراج ویژگی معروف، الگوریتم SIRF است که با هدف رفع مشکل الگوریتم هریس ایجاد شد. در این الگوریتم، برعکس هریس در برابر تغییر مقیاس مقاوم است. همچنین علاوه بر مقاومت در برابر تغییر مقیاس، در برابر تغییرات چرخش و روشنایی و نویز نیز مقاوم است.

فهرست مطالب

۱-مقدمه

با پیشرفت فناوری دستگاه های تصویربرداری و افزایش کیفیت تصاویر و همچنین افزایش چشم‌گیر حجم داده ها، ذخیره سازی تصویر، یکی از چالش های مهم در ذخیره سازی محسوب می‌شود. از این رو با یافتن، استخراج و ذخیره نقاطی از تصویر که بیانگر ویژگی های اصلی از تصویر هستند، می‌توان به جای خود تصویر، ویژگی های آن را ذخیره نمود و از این نقاط ویژگی برای ذخیره سازی و شناسایی تصویر استفاده نمود. الگوریتم های متنوعی برای یافتن و استخراج ویژگی معرفی شده اند.

در این گزارش، با نقاط ویژگی و یکی از الگوریتم های استخراج ویژگی آشنا می‌شویم.

۲- توضیحات فنی

۲-۱- Harris Corner Detector

۲-۱-۱- بخش اول

ایده اولیه گوشه یاب هریس به این صورت است که نقاط را با بررسی از طریق یک پنجره کوچک، تشخیص می‌دهیم.

یک گوشه این خاصیت را دارد که اگر پنجره یاد شده را در هر راستایی جابه‌جا کنیم، تغییرات بزرگی را در شدت روشنایی خواهیم داشت. برای یک ناحیه مسطح، جابه‌جا کردن پنجره در هر راستا تغییر چندانی در پی ندارد. برای یک لبه، این تغییرات تنها در یک راستا (راستای لبه)، بزرگ خواهد بود و برای یک گوشه، همانطور که بیان شد، جابجایی پنجره در هر راستا، تغییرات قابل ملاحظه‌ای خواهد داشت. به بیان ریاضی، گرادیان یک ناحیه مسطح کوچک است. گرادیان یک لبه افقی در راستای عمودی و گرادیان یک لبه عمودی در راستای افقی بزرگ است. همچنین برای یک گوشه، گرادیان در هر دو راستا، بزرگ است.

بادر نظر گرفتن توضیحات فوق می‌توان تابع E(u,v) را، که مقدار تغییرات گرادیان را برای پنجره شیفت داده شده به اندازه (u,v) محاسبه می‌کند، به صورت زیر معرفی کرد:

فرمول (۱)

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

که w(x,y) تابع پنجره، I(x,y) شدت روشنایی و I(x+u, y+v) شدت روشنایی شیفت داده شده به اندازه (u,v) است.

با بسط I(x,y) در بسط سری تیلور، برای شیفت کوچک (u,v)، یک تخمین دو خطی به جای تخمین درجه دو داریم:

فرمول (۲)

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

که M یک ماتریس ۲ در ۲ به نام ماتریس ممان دوم یا تانسور ساختار[1] است که از مشتقات تصویر محاسبه می‌شود:

فرمول (۳)

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

به جز مقادیر ویژه ماتریس M، برای تابع پاسخ گوشه می‌توان معیار های دیگری نیز در نظر گرفت. یکی از این معیار ها، ماتریس R است که به صورت زیر تعریف می‌شود:

فرمول (۴)

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

[1] Structure tensor

در این رابطه نیازی به محاسبه مقادیر ویژه نیست. ثابت α، ثابت تجربی است که مقدار آن بین ۰,۰۴ و ۰,۰۶ انتخاب می‌شود. برای یک گوشه، R بزرگتر از صفر است. برای یک لبه، R منفی است و برای یک ناحیه مسطح، اندازه R مقدار کوچکی است.

بنابراین با جمع بندی آنچه گفته شد، مراحل در الگوریتم گوشه یابی هریس به صورت زیر است:

۱. محاسبه مشتقات گوسی در هر پیکسل

۲. محاسبه ماتریس ممان دوم M در یک پنجره گوسی حول هر پیکسل

۳. محاسبه تابع پاسخ گوشه R

۴. آستانه گیری بر روی ماتریس R

۵. یافتن ماکزیمم محلی تابع پاسخ (non-max suppression)

خروجی الگوریتم فوق، مجموعه ای از نقاط است که گوشه های تصویر هستند.

بنابراین ما در پیاده سازی الگوریتم گوشه یاب هریس ابتدا تصویر خاکستری را به کمک فیلتر گوسی، هموار کرده و سپس در ادامه به کمک فیلتر سوبل در راستای افقی و عمودی، درایه های ماتریس M را می‌سازیم.

سپس با استفاده از دترمینان و اثر ماتریس M و ثابت α، ماتریس R را می‌سازیم. در ادامه با انتخاب یک آستانه مناسب و نرمال سازی ماتریس R، آن را آستانه گیری می‌کنیم. با این کار نقاطی به دست می‌آوریم که کاندید نقطه ویژگی بودن هستند. سپس با استفاده از non-max suppression، گوشه ها را یافته و این نقاط را روی تصویر رنگی مشخص می‌کنیم.

۲-۲- Scene stitching with SIFT/SURF features

۲-۲-۱- بخش اول

الگوریتم SIFT بعد از الگوریتم گوشه‌یاب هریس معرفی شد. مشکل الگوریتم گوشه‌یاب هریس این بود که در برابر تغییرات مقیاس مقاوم نبود، در نتیجه الگوریتم SIFT معرفی شد. این الگوریتم نسبت به تغییر مقیاس، چرخش، روشنایی و نویز مقاوم است. detector مربوط به این الگوریتم، ابتدا فضای مقیاس را تولید میکند به این صورت که در هر octave، روی تصویر با سایز یکسان فیلتر گوسی با سیگما های مختلف اعمال میشود و پس از چند مرحله اعمال فیلتر گوسی، طبق اصل شانون چون حداکثر فرکانس تصویر کاهش میابد پس اندازه تصویر را کاهش میدهیم. برای یافتن نقاط ویژگی، این الگوریتم[1] LoG را برای تصاویر داخل هر octave محاسبه میکند. چون بار محاسباتی LoG زیاد می توان آن را با[2] DoG تخمین زد.

یک نقطه را نقطه ویژگی تعریف میکنیم اگر آن نقطه در تمام DoG ها در همسایگی ۳ در ۳ خود، یک نقطه اکسترما (ماکزیمم یا مینیمم محلی) باشد. سپس برای انتخاب نقاط ویژگی نهایی، نقاطی که اندازه DoG آنها کمتر از ۰,۰۳ باشند و همچنین نقاط لبه حذف میشوند.

در descriptor این الگوریتم ابتدا یک جهت اصلی برای تصویر انتخاب میکنیم. سپس بر اساس این جهت اصلی، توصیفگر مربوط به هر نقطه ویژگی را ایجاد میکنیم. در نهایت الگوریتم SIFT برای هر نقطه ویژگی یک بردار ۱۲۸ تایی در نظر میگیرد و برای تطابق دو نقطه از دو تصویر، از این بردار استفاده میشود و دو نقطه مطابق بر هم خواهند بود اگر بردار انها مشابه هم باشد.

در این بخش ما از الگوریتم SIFT برای پیدا کردن نقاط ویژگی دو تصویر و تطابق دو به دوی این ویژگی ها استفاده کرده ایم.

در این بخش، متد show_matched_features پیاده شده است که این متد دو تصویر را به عنوان ورودی میگیرد و در آن ابتدا نقاط ویژگی SIFT را استخراج میکنیم، سپس feature های همسایه را استخراج میکنیم و پس از آن بین feature های

[1] Laplacian of Gaussian
[2] Difference of Gaussian

استخراج شده برای دو تصویر یک نگاشت پیدا میکنیم و ویژگی هایی از دو تصویر که با یکدیگر منطبق هستند را در تصویر خروجی با یک خط به هم متصل میکنیم.

۲-۲-۲- بخش دوم

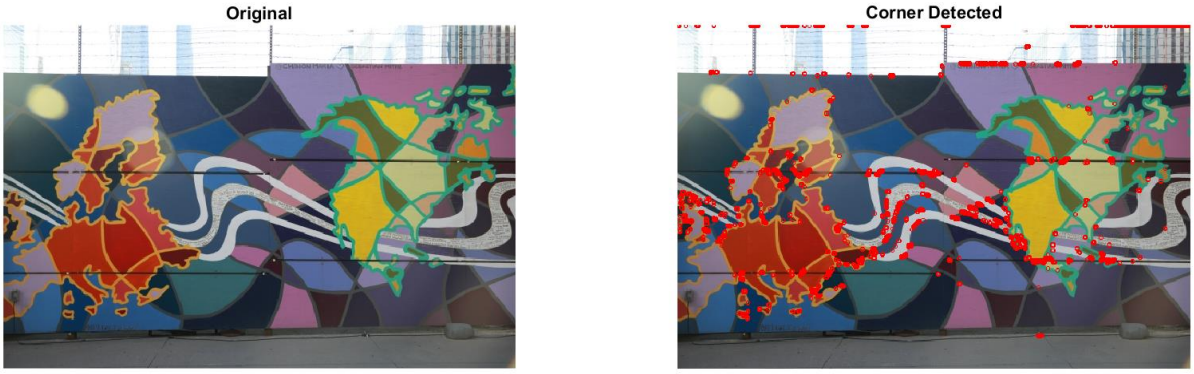
کد بخش قبل را روی ۳ تصویر که با دوربین گوشی از زوایای مختلف گرفته شده اجرا کرده ایم که نتایج آن را می‌توانید در بخش بررسی نتایج مشاهده کنید.

۳- بررسی نتایج

۳-۱- Harris Corner Detector

۳-۱-۱- بخش اول

همانطور که در تصویر زیر مشاهده میکنید، تصویر زیر نقاط گوشه را به خوبی نمایش میدهد:

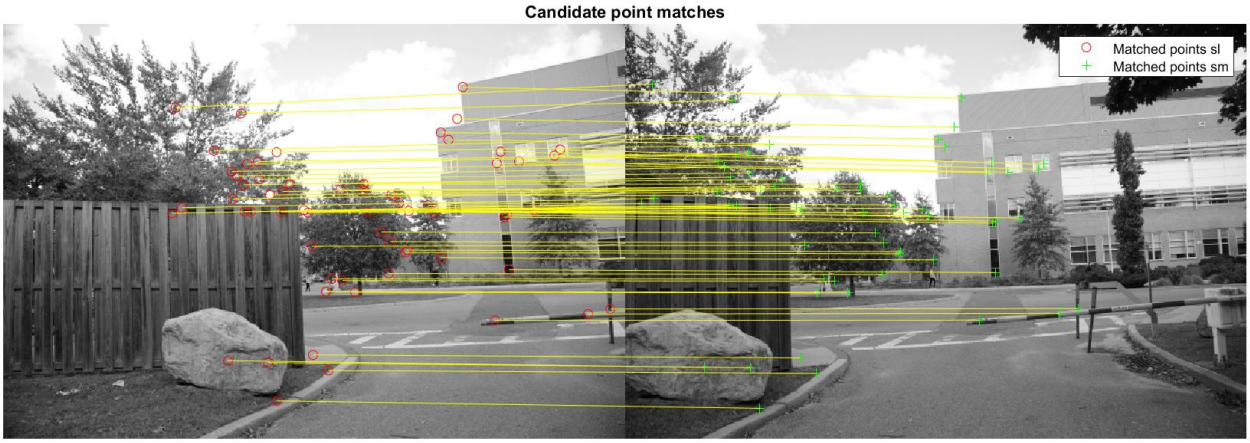


تصویر ۱- نقاط گوشه با آلفا برابر با ۰،۰۵ و استانه برابر با ۰،۵

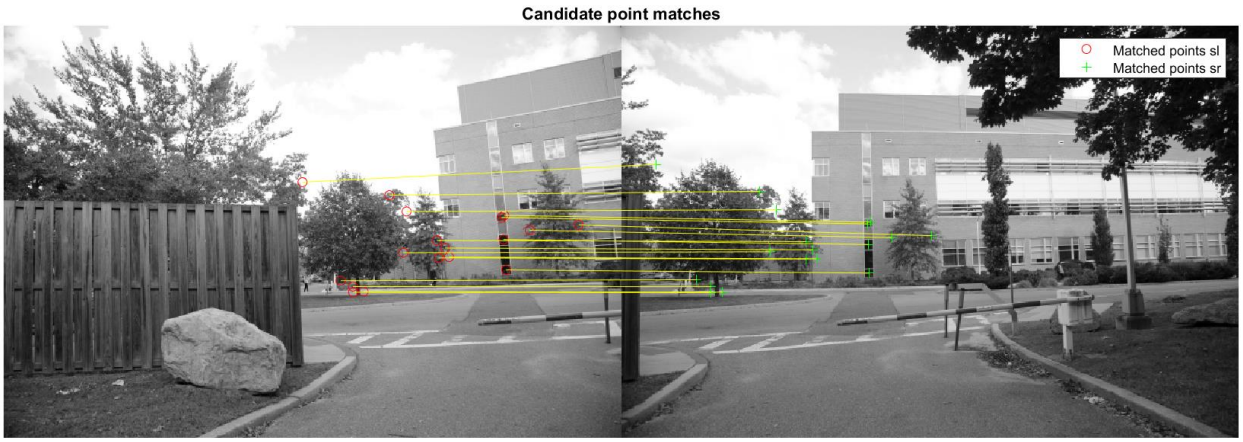
۳-۲- Scene stitching with SIFT/SURF features

۳-۲-۱- بخش اول

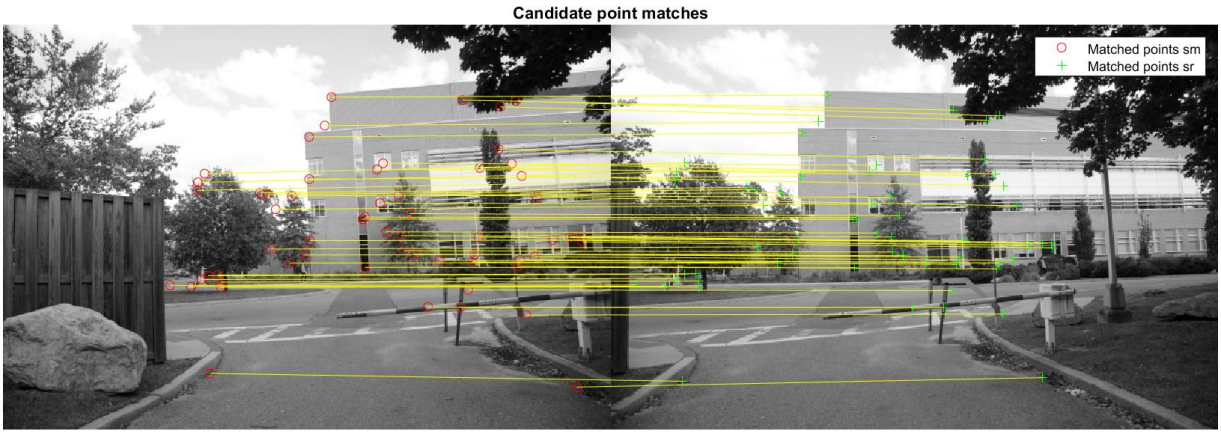
همانطور که در تصاویر زیر مشاهده میکنید، ویژگی های SIFT دو تصویر که بر هم منطبق هستند با یک خط به هم وصل شده اند. تعداد ویژگی های منطبق بر هم برای هر دو تصویر متفاوت می‌باشد و برای دو تصویر ممکن است ویژگی های مشابه بیشتر و برای دو تصویر دیگر از همان صحنه ولی با زاویه دید متفاوت، ویژگی های مشابه کمتری یافت شود.



تصویر ۲- ویژگی های SIFT منطبق بر هم از دو تصویر sl و sm



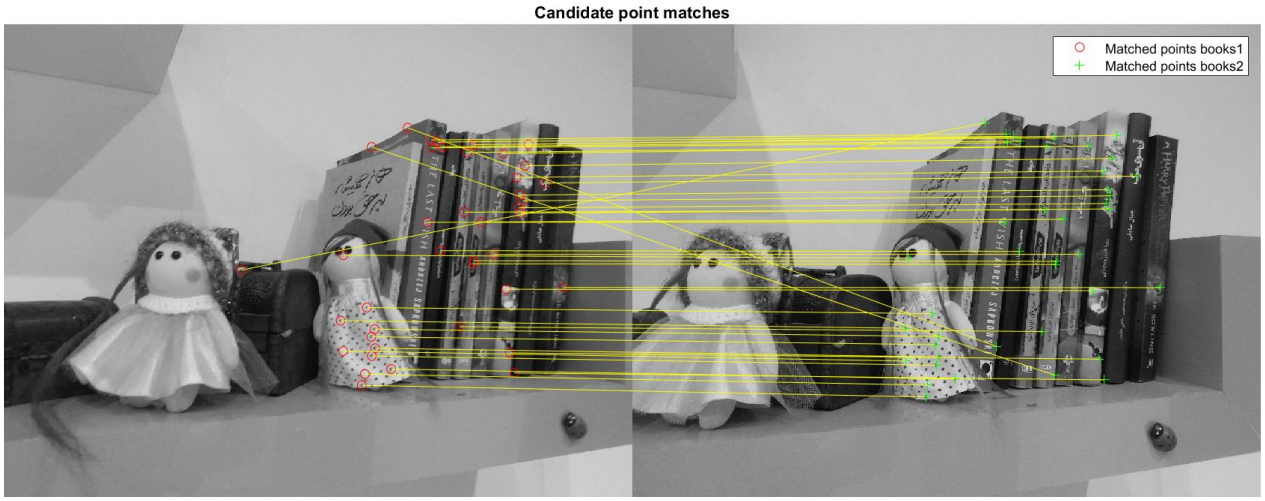
تصویر ۳- ویژگی های SIFT منطبق بر هم از دو تصویر sl و sr



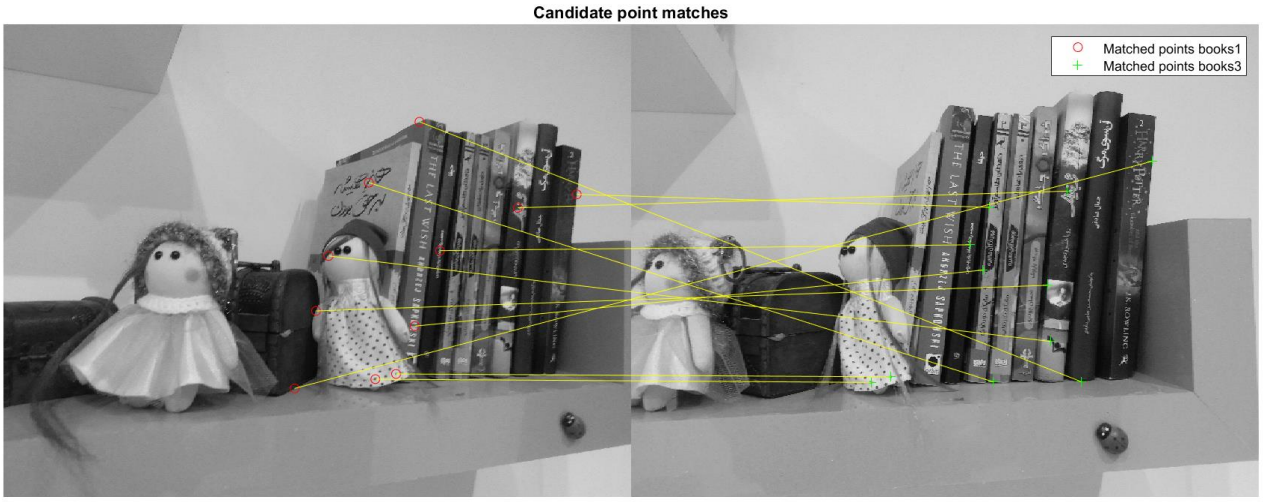
تصویر ۴- ویژگی های SIFT منطبق بر هم از دو تصویر sr و sm

۳-۲-۲- بخش دوم

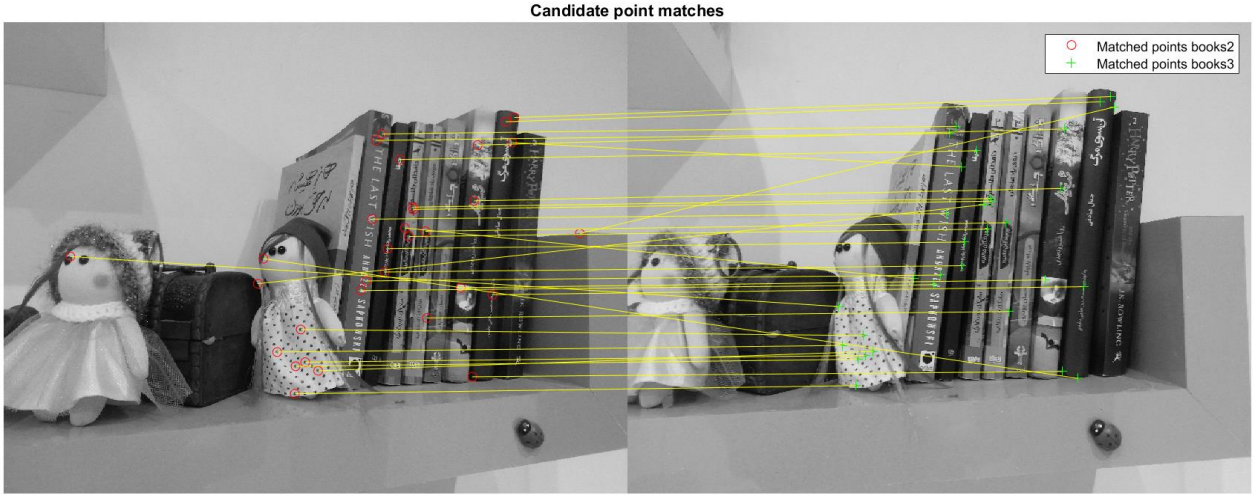
در این بخش ۳ تصویر با زوایای مختلف از یک قفسه کتاب گرفته شده است. همانطور که مشاهده میکنید برای دو به دوی این تصاویر ویژگی های مشابه نمایش داده شده است. نکته لازم به ذکر این است که همانطور که میبینیم برخی نقاط ویژگی به اشتباه مشابه هم تشخیص داده شده اند که این نشان میدهد که الگوریتم SIFT دارای خطا است و دقت آن ۱۰۰ درصد نیست.



تصویر ۵- تطابق ویژگی های SIFT در دو تصویر books1 و books2



تصویر ۶- تطابق ویژگی های SIFT در دو تصویر books1 و books3



تصویر ۷- تطابق ویژگی های SIFT در دو تصویر books2 و books3

۴- پیوست

۴-۱- Harris Corner Detector

۴-۱-۱- بخش اول

کد main:

```
harris_img = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\harris.JPG");
redChannel = harris_img(:, :, 1);
greenChannel = harris_img(:, :, 2);
blueChannel = harris_img(:, :, 3);
harris_img_rgb = cat(3, redChannel, greenChannel, blueChannel);
harris_img_gray = double(rgb2gray(harris_img));

% Compute Gaussian derivatives at each pixel
guassian_kernel = [1 2 1; 2 4 2; 1 2 1] * 1/16;
filtered_harris_img = conv2(harris_img_gray, guassian_kernel, 'same');

% Compute second moment matrix M in a Gaussian window around each pixel
sobel_x_kernel = [-1 0 1; -2 0 2; -1 0 1];
sobel_y_kernel = [1 2 1; 0 0 0; -1 -2 -1];
```

```
Ix = conv2(filtered_harris_img, sobel_x_kernel, 'same');
Iy = conv2(filtered_harris_img, sobel_y_kernel, 'same');

Ix2 = Ix.^2;
Iy2 = Iy.^2;
IxIy = Ix.*Iy;

g_Ix2 = conv2(Ix2, guassian_kernel, 'same');
g_Iy2 = conv2(Iy2, guassian_kernel, 'same');
g_IxIy = conv2(IxIy, guassian_kernel, 'same');
alpha = 0.05;
R_matrix = (g_Ix2 .* g_Iy2) - (g_IxIy.^2) - (alpha * (square(g_Ix2 + g_Iy2)));
min(R_matrix(:))
max(R_matrix(:))
%R_matrix = (R_matrix - min(R_matrix(:)))./(max(R_matrix(:))-min(R_matrix(:)));
R_matrix = normalize(R_matrix,'range', [0,1]);
% threshold R_matrix
threshold = 0.8;
[row, col] = find(R_matrix >= threshold);
key_points = cat(2, row, col);
% Find local maxima of response function (non-maximum suppression)
corner_points = [0 0];
for kp=1:length(key_points)
    row = key_points(kp,1);
    col = key_points(kp,2);
    if R_matrix(row, col) > 0
        corner_points = cat(1, corner_points, [col row]);
    end
end
corner_points = corner_points(2:length(corner_points),:);
% draw corners on image
corner_detected_image = insertMarker(harris_img_rgb, corner_points, 'o', 'Color',
'red','size',2);
figure;
subplot(1, 2, 1); imshow(harris_img_rgb); title("Original");
subplot(1, 2, 2); imshow(corner_detected_image); title("Corner Detected");
```

Scene stitching with SIFT/SURF features ۴-۲-

۴-۲-۱- بخش اول

کد main:

```
img1 = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\sl.jpg");
img2 = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\sm.jpg");
img3 = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\sr.jpg");

show_matched_features(rgb2gray(img1), rgb2gray(img2), "Matched points sl",
"Matched points sm");
show_matched_features(rgb2gray(img1), rgb2gray(img3), "Matched points sl",
"Matched points sr");
show_matched_features(rgb2gray(img2), rgb2gray(img3), "Matched points sm",
"Matched points sr");
```

کد متد show_mathced_features:

```
function show_matched_features(img1, img2, title1, title2)
% find SIFT features
img1_points = detectSIFTFeatures(img1,"EdgeThreshold",2);
img2_points = detectSIFTFeatures(img2,"EdgeThreshold",2);

% Extract the neighborhood features
[features1, valid_points1] = extractFeatures(img1, img1_points);
[features2, valid_points2] = extractFeatures(img2, img2_points);

% Retrieve the locations of matched points
indexPairs = matchFeatures(features1, features2);
matchedPoints1 = valid_points1(indexPairs(:,1),:);
matchedPoints2 = valid_points2(indexPairs(:,2),:);

% Display the matching points
figure; ax = axes;
showMatchedFeatures(img1, img2, matchedPoints1,
matchedPoints2,"montag",Parent=ax);
title(ax,"Candidate point matches");
legend(ax, title1, title2);
end
```

۴-۲-۲- بخش دوم

کد main:

```
img1 = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\books1.jpg");
img2 = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\books2.jpg");
img3 = imread("D:\Dars\Masters\digital image
processing\Homeworks\Images\7\books3.jpg");

show_matched_features(rgb2gray(img1), rgb2gray(img2), "Matched points books1",
"Matched points books2");
show_matched_features(rgb2gray(img1), rgb2gray(img3), "Matched points books1",
"Matched points books3");
show_matched_features(rgb2gray(img2), rgb2gray(img3), "Matched points books2",
"Matched points books3");
```