Image Alignment and Stitching Algorithms Behind Google Street View

Shuai Yuan
Electrical and Computer Engineering,
University of Arizona,
Tucson, AZ, USA
yuanshine@email.arizona.edu

1. Introduction

Image alignment and stitching is one of the most important technologies in Google Street View. It also applies generally on the region of image processing and computer vision. In general, all algorithms of image alignment and stitching can be classified into tow categories: pixel-based and feature-based image registration. Since the latter method performs better in terms of robustness, in this report, I will mainly focus on it.

To design algorithms for image alignment and stitching, three elements, which are filter design, parameter estimation and spatial geometric transformation, are supposed to be considered. Section 2 will discuss a traditional algorithm,[1] while section 3 will introduce an advanced method, which is more robust to camera zoom, orientation of the input images.[5] Both of them are based on feature-based image registration and subjective to those three designing considerations.

2. General Feature-Based Image Registration

The basic idea of feature-based image registration is to extract features from each image, to initialize feature correspondences, to find a set of high-accuracy alignment, and to estimate the geometric transformation between images. Traditionally, Harris corner detection algorithm is used to extract corners as key points, which are then matched according to windowed spatially varying weights estimate function. For image alignment, RANSAC is widely used and turned out to be an effective algorithm. In the last step, we need to implement homography update to finish image stitching. [1]

2.1 Harris Corner Detection Algorithm

Harris and Stephens proposed a Gaussian wighting function approach to find corners as key points in image. The Hessian and eigenvalue images can be efficiently evaluated using a sequence of filters and algebraic operations,

$$G_x(x,y) = \frac{\partial}{\partial x} G(x, y, \sigma_d) \otimes I(x, y)$$

$$G_y(x,y) = \frac{\partial}{\partial y} G(x, y, \sigma_d) \otimes I(x, y)$$

$$B(x, y) = \begin{bmatrix} G_x^2(x, y) & G_x(x, y)G_y(x, y) \\ G_x(x, y)G_y(x, y) & G_y^2(x, y) \end{bmatrix}$$

$$A(x, y) = G(x, y, \sigma_i) \otimes B(x, y)$$
$$\frac{\det(A(x, y))}{tr(A(x, y))} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$$

where * is defined as convolution, I(x, y) is the image to be processed, G(x, y, σ) is Gaussian distributed function with mean 1 and variance σ, and det and tr represents determinant and trace of matrix, respectively. Finally, corners are determined by the points whose value of det(•)/tr(•) is greater than a threshold. These corners are extracted as key points. [1]

2.2 Windowed Spatially Varying Weights Estimator

The least squares estimator is a measurement of errors between features of a pair of images. The pair of pixels with the least error is selected as matched feature. In some cases, some of the pixels being compared may lie outside the original image boundaries. Thus, it is necessary to associate a spatially varying per-pixel weight with each of the two images to be matched. In summary, windowed spatially varying weights estimator can be written as below,

$$\mathrm{E}(\Delta x, \Delta y) = \sum_{i=1}^{I} \sum_{j=1}^{J} w_1(x, y)\, w_2(x+\Delta x, y+\Delta y)\, [I_2(x_i+\Delta x, y_i+\Delta y) - I_1(x_i, y_i)]^2$$

where the weighting functions w are zero outside the valid ranges of the images, (Δx, Δy) is the displacement between two images. [1]

2.3 RANSAC Algorithm

RANSAC is short for random sampling consensus and provides a solution to the problem that finds consistent points with some particular motion estimate. RANSAC algorithm starts a set of matched features and ends up with a set of inliers. In detail, RANSAC algorithm is initialized with a randomly selected subset of correspondences to compute a motion estimate. Next, determine how many points from the set of all points fit with a predefined tolerance by solving for the parameters o the model. If the fraction of the number of inliers over the total number points in the set exceeds a specific threshold, re-estimate the model parameters using all the identified inliers and terminate. Otherwise, repeat the steps stated above. The number of iterations N is decided by the following formula,

$$N = \frac{\log(1-p)}{\log(1-u^m)}$$

where p is the probability that at least one of the sets of random samples does not include an outlier, u is the probability that any selected pixel is inlier, and m is the minimum number of points. [2]

2.4 Homography Update

Homography is also known as perspective transform. Homography update is the last step in image alignment and stitching, which transforms each image from 2D to 3D coordinates. In this project, image series are collected by pure rotation of camera. Therefore, in this section, only this special case is considered. Assume that all points in the image set are infinite far from the camera, then the homography matrix can be written as follow,

$$H = \begin{bmatrix} \cos\theta & 0 & f\sin\theta \\ 0 & 1 & 0 \\ -\frac{1}{f}\sin\theta & 0 & \cos\theta \end{bmatrix}$$

where f is the focal length of camera, and θ is the rotation angels of camera. With this matrix, we can write the transformation of points from 2D to 3D coordinate systems as,

$$\begin{bmatrix} x_1 \\ y_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & f\sin\theta \\ 0 & 1 & 0 \\ -\frac{1}{f}\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

3. Invariant Feature-Based Image Registration

The basic steps in the invariant feature-based image registration is similar to that in the previous section. However, they differ from some specific techniques. Invariant feature-based registration detects and extracts features based on SIFT algorithm and then matches them by k-d tree. RANSAC is still applied to find inliers of each image. The largest improvement is that bundle adjustment is introduced to perform global adjustment, which will focus on minimizing the error between all pairs of images instead of only one pair.

3.1 SIFT Algorithm

SIFT features are located at scale-space maxima/minima of a difference of Gaussian function. At each location, a characteristic scale and orientation is established, which makes extracted features invariant under rotation and scaling changes. There are four major steps in the SIFT algorithm: scale-space extrema detection, key points localization, orientation assignment and key point descriptor. Similar to section 2.1, scale space extrema detection is implemented by using a difference of Gaussian distribution function to identify key points. The processed image L(x, y, σ) is denoted as

$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x,y)$$

Next, scale-space extrema D(x, y, σ) is calculated by

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Accurate key point localization D(x', y', σ) is firstly calculated from D(x, y, σ),

$$D(x', y', \sigma) = D(x, y, \sigma) - \frac{1}{2} \frac{\partial D^T(x, y, \sigma)}{\partial D(x, y)} \frac{\partial^2 D^{-1}(x, y, \sigma)}{\partial D^2(x, y)} \frac{\partial D(x, y, \sigma)}{\partial D(x, y)}$$

and if D(x', y', σ) is greater than a threshold, (x', y') is kept as key points. While if D(x', y', σ) is less than that threshold, (x', y') is discarded. Then, each accurate key point is assigned with scale and orientation according to

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \arctan \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}$$

The local image gradients are measured at the selected scale in the region around each key point. These are transformed into a representation that allows for significant level of local shape distortion and change in illumination. [6]

3.2 k-d Trees

k-d tree, also known as k-dimensional tree is a space partitioning data structure in computer science. It is usually used to partition data set into a k dimensional space or k nearest neighbors. When this theory applies to invariant feature-based image registration, the data set is features extracted from all images, and then match each features to its k nearest neighbors. As one feature moves down the tree, it cycles through all dimensions used to select a specific plane. And then matched features are inserted by selecting the median of the feature descriptors being put into subtree. Since the implementation of it requires data structure background, I just introduce it in this section but still use the matching approach in section 2.2. [7]

4. Result

4.1 Feature Extraction

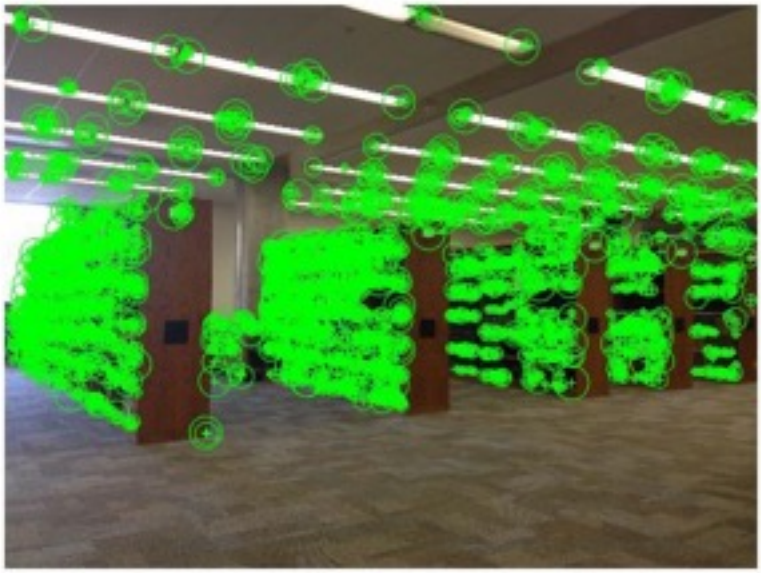
Figure 1(a) is the image to be processed by Harris corner detector algorithm or SIFT algorithm. In Figure 1(b), the green marks are key points extracted by Harris corner detector algorithm. In Figure 1(c) the green marks are key points extracted by SIFT algorithm. The circle outside cross denote the assigned scale of the key points.

By comparison, the number of key points in figure 4(a) is 1528, which is less than that in figure 4(b), 3079. Meanwhile, extracted feature in figure 4(a) is not so accurate as that in figure 4(b). For example, the two crosses on the ground are obviously not part of features.

4.2 Feature Matching

(a)

(b)

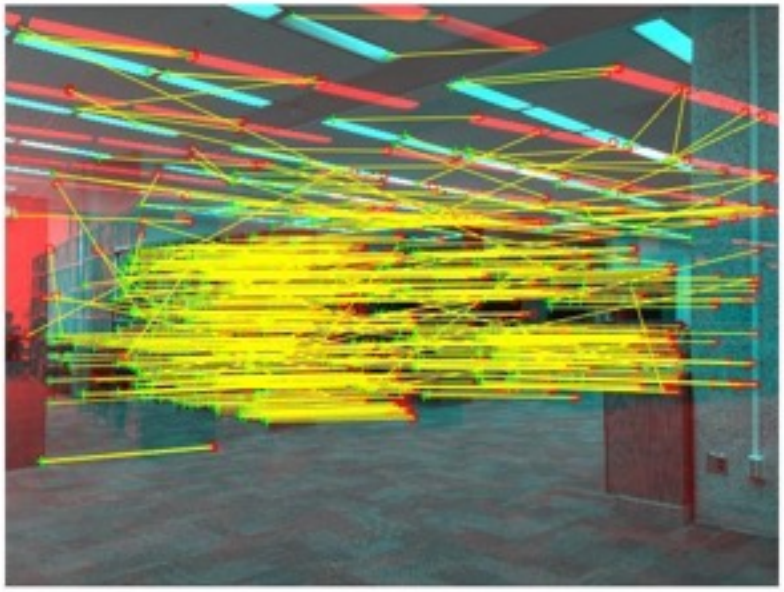


(c)
Figure 1

（a）

(b)

(c)



(d)
Figure 2

Figure 2(a) and 2(b) are two images to be matched. Figure 2(c) and 2(d) show feature matching that results from key point extracted by Harris corner detector algorithm and SIFT algorithm, respectively. In the figure, green cross denotes feature original in Figure 2(a), red dot denotes feature original in Figure 2(b), and the yellow lines are displacement of a pair of matched features. Figure 2(c) and 2(d) indicate that SIFT algorithm outperforms Harris corner detector algorithm again.

4.3 Image Alignment

To investigate RANSAC algorithm, Figure 2(a) and 2(b) are still used as original images. Figure 3(a) and 3(b) show image alignment that results from key point extracted by Harris corner detector algorithm and SIFT algorithm, respectively. In the figure, red circle denotes inlier original in Figure 2(a), green cross denotes inlier original in Figure 2(b), and the yellow lines are displacement of extracted inliers by RANSAC algorithm. Figure 3(b) has more inliers than Figure 3(a), which will lead to a more accurate results. Furthermore, RANSAC algorithm eliminates the pairs of matched features which will potentially lead to errors. After implementing RANSAC algorithm, the number of matching features reduce from 590 to 207 for SIFT algorithm, while they reduce from 10 to 8 for Harris corner detector algorithm.

4.4 Image Stitching

4.4.1 Image Stitching Under Variant Orientations

Figure 4(a), 4(b) and 4(c) are a sequence of images. Figure 4(b) rotates 10° based on Figure 4(a), and Figure 4(c) rotates 20° based on Figure 4(b). Figure 5(d) is stitched by general feature based registration. Figure 5(e) is stitched by invariant feature based registration. Both of them are almost perfect.

4.4.2 Image Stitching Under Variant Scales

Figure 5(a), 5(b) and 5(c) are images to be stitched. As figures indicate, they are variable in terms of size which are selected randomly. Figure 5(d) is stitched by general feature based registration. The result is nothing but a bug. Figure 5(e) is stitched by invariant feature based registration, which is almost perfect. In conclusion, invariant feature based registration is more robust when the scale of images is not unified.

4.4.3 Panaromas

Figure 6 shows a stitched panaromas that samples from 6 independent images.

5 Conclusion

In this project, I have learnt fundamental theories in image alignment and stitching, from introductory level to advanced level. I primarily investigated two dominant algorithms in

(a)

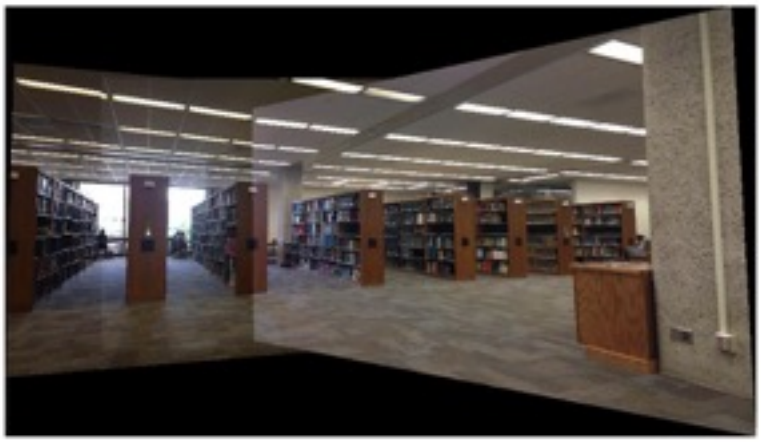
(b)
Figure 3

computer vision industry. I used computer vision toolbox predefined in MATLAB to develop program. I also coded on some fundament functions by myself, which can replace some predefined functions. All of them will be presented in appendix.

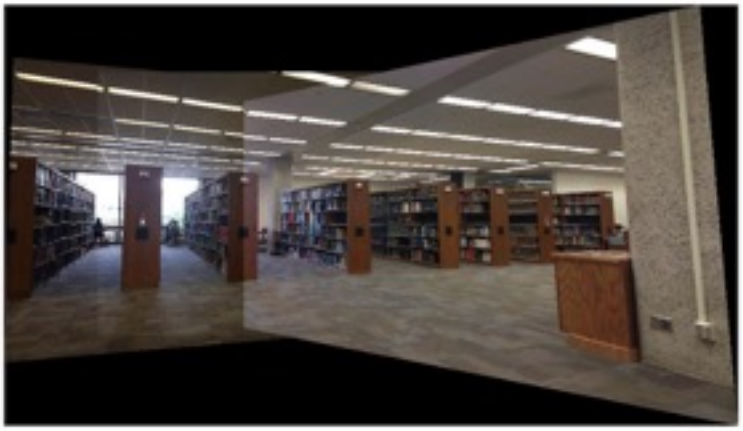
(a)

(b)

(c)



(d)



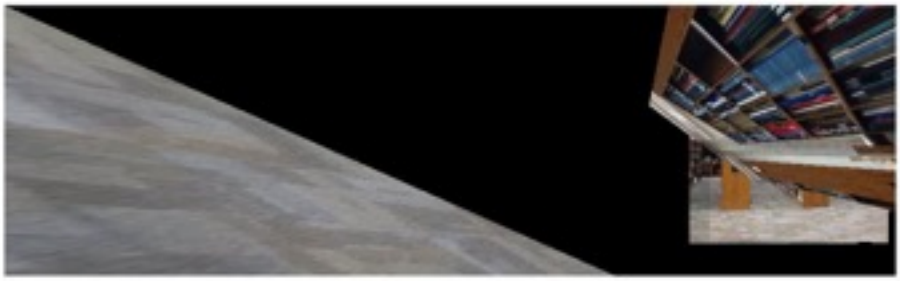
(e)

Figure 4

(a)

(b)

(c)



(d)

(e)
Figure 5

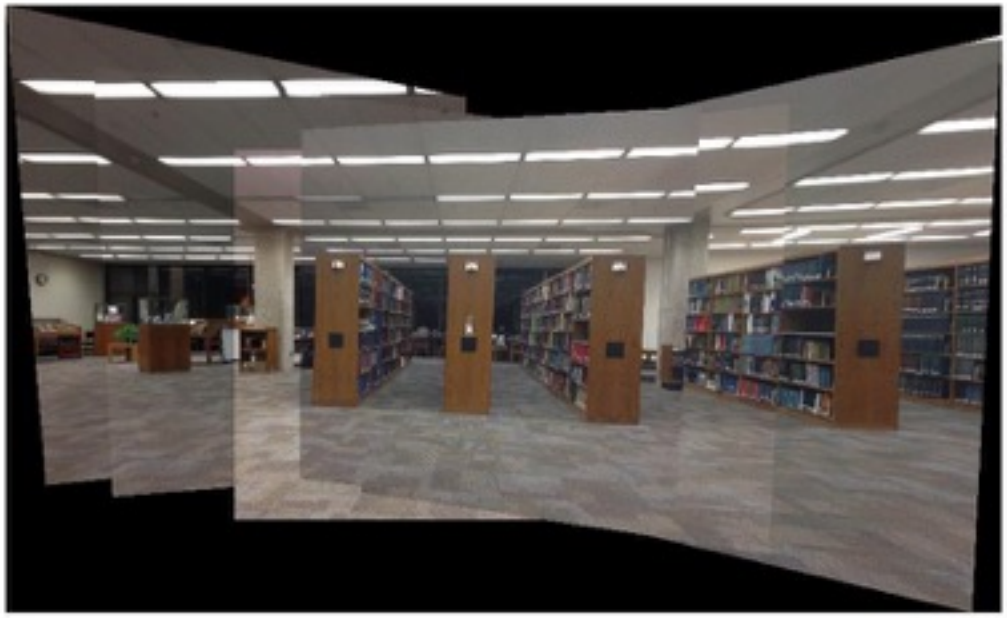


Figure 6

By comparison, invariant feature based image registration outperforms general feature image registration. Because 1) invariant feature based image registration can extract

more scale and orientation invariant features; 2) invariant feature based image can registration matches features more accurate than general feature image registration. 3) invariant feature based image show good robustness property under variant scales. However, both of them performs well under variant orientations, which is a little different from the materials I read [5].

I will continue working on this project after this semester. I will implement k-d trees algorithm to accelerate computing. Thanks for Professor Rodriguez.

Reference

[1] R. Szeliski. Image Alignment and Stitching: A Tutorial. *Computer Graphics and Vision* Vol. 2, No.1 (2006) 1-104
[2] C.G. Harris and M.J. Stephens. "A combined corner and edge detector". *Proceedings Fourth Alvey Vision Conference*, Manchester, pp 147-151, 1988.
[3] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6): 381-395, 1981
[4] D. Kriegman. Homography Estimation.
[5] M. Brown and D.G. Lowe. 2007. Automatic Panoramic Image Stitching using Invariant Features. *Int. J. Comput. Vision 74, 1* (August 2007), 59-73.
[6] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 2004.
[7] A.W. Moore. AN intriducory tutorial on k_d trees.

Appendix

```
clear all;
clc;

buildingScene =
imageSet({'02.jpg','03.jpg','04.jpg','05.jpg','06.jpg','07.jpg','08.jpg','09.
jpg'});

%montage(buildingScene.ImageLocation)

I = read(buildingScene, 1);

grayImage = rgb2gray(I);
points = detectSURFFeatures(grayImage);
[features, points] = extractFeatures(grayImage, points);

tforms(buildingScene.Count) = projective2d(eye(3));

for n = 2:buildingScene.Count

    pointsPrevious = points;
    featuresPrevious = features;
```

```
    I = read(buildingScene, n);

    grayImage = rgb2gray(I);
    points = detectSURFFeatures(grayImage);
    [features, points] = extractFeatures(grayImage, points);

    indexPairs = matchFeatures(features, featuresPrevious, 'Unique', true);

    matchedPoints = points(indexPairs(:,1), :);
    matchedPointsPrev = pointsPrevious(indexPairs(:,2), :);

    tforms(n) = estimateGeometricTransform(matchedPoints,
matchedPointsPrev,...
        'projective', 'Confidence', 99.9, 'MaxNumTrials', 2000);

    tforms(n).T = tforms(n-1).T * tforms(n).T;
end

imageSize = size(I);

for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(2)], [1
imageSize(1)]);
end

avgXLim = mean(xlim, 2);

[~, idx] = sort(avgXLim);

centerIdx = floor((numel(tforms)+1)/2);

centerImageIdx = idx(centerIdx);

Tinv = invert(tforms(centerImageIdx));

for i = 1:numel(tforms)
    tforms(i).T = Tinv.T * tforms(i).T;
end

for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(2)], [1
imageSize(1)]);
end

xMin = min([1; xlim(:)]);
xMax = max([imageSize(2); xlim(:)]);

yMin = min([1; ylim(:)]);
yMax = max([imageSize(1); ylim(:)]);

width  = round(xMax - xMin);
height = round(yMax - yMin);

panorama = zeros([height width 3], 'like', I);
```

```
blender = vision.AlphaBlender('Operation', 'Binary mask','MaskSource', 'Input
port');

xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);

for i = 1:buildingScene.Count

    I = read(buildingScene, i);

    warpedImage = imwarp(I, tforms(i), 'OutputView', panoramaView);

    warpedMask = imwarp(ones(size(I(:,:,1))), tforms(i), 'OutputView',
panoramaView);

    warpedMask = warpedMask >= 1;

    panorama = step(blender, panorama, warpedImage, warpedMask);
end

figure
imshow(panorama)

clear all;
clc;

I1 = rgb2gray(imread('IMG_04.jpg'));
I2 = rgb2gray(imread('IMG_05.jpg'));

p1_0 = detectSURFFeatures(I1);
p2_0 = detectSURFFeatures(I2);

[f1,p1] = extractFeatures(I1,p1_0);
[f2,p2] = extractFeatures(I2,p2_0);
index = matchFeatures(f1,f2);
mp1 = p1(index(:,1),:);
mp2 = p2(index(:,2),:);
size(index)
figure;
showMatchedFeatures(I1,I2,mp1,mp2);

[tform,inliers1,inliers0] = estimateGeometricTransform(mp1,
mp2,'projective');
size(inliers0)
size(inliers1)
figure;
showMatchedFeatures(I1,I2,inliers0,inliers1);

function [cim, r, c] = Harris(im, sigma, thresh, radius)

% This function is to implement harris corner detector to extract key points.
```

```
% im = image to be processed.
% sigma = standard deviation of Gaussian filter.
% threshold = It is a conner if its value is greater than threshold.
% radius = radius of region in non-maximal suppression.

% cim = proecessed image marking with corners.
% r = row coordinates of key points.
% c = column coordinates of key points.

    p = [0.030320  0.249724  0.439911  0.249724  0.030320];
    d = [0.104550  0.292315  0.000000 -0.292315 -0.104550];

    Ix = conv2(p, d, im, 'same');
    Iy = conv2(d, p, im, 'same');

    f_size = ceil(6*sigma);
    if ~mod(f_size,2)
       f_size = f_size+1;
    end

    h = fspecial('gaussian', [f_size f_size], sigma);
    Ix2 = filter2(h,Ix.^2);
    Iy2 = filter2(h,Iy.^2);
    Ixy = filter2(h,Ix.*Iy);

    cim = (Ix2.*Iy2 - Ixy.^2)./(Ix2 + Iy2 + eps);

    D = 2*radius+1;
    mx = ordfilt2(cim, D^2,ones(D));

    bordermask = zeros(size(cim));
    bordermask(radius+1:end-radius, radius+1:end-radius) = 1;

    cimmx = (cim==mx) & (cim>thresh) & bordermask;
    [r,c] = find(cimmx);
    [r,c] = NonMaxSuppression(cim, radius, thres
```