AGH University of Science and Technology

Department of Telecommunication

Corner Extraction

Introduction to image processing using Matlab

Autor: ***Zbigniew Hulicki*** ©

Kraków, *2017*

1. Theoretical fundamentals.

Corner Extraction – (Corner detection) is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image. Corner detection is frequently used in motion detection, image registration, video tracking, image mosaicing, panorama stitching, 3D modelling and object recognition. Corner detection overlaps with the topic of interest point detection.

A corner can be defined as the intersection of two edges. A corner can also be defined as a point for which there are two dominant and different edge directions in a local neighborhood of the point.

An interest point is a point in an image that has a well-defined position and can be robustly detected. This means that an interest point can be a corner but it can also be, for example, an isolated point of local intensity maximum or minimum, line endings, or a point on a curve where the curvature is locally maximal.

In practice, most so-called corner detection methods detect interest points in general, and in fact, the term "corner" and "interest point" are used more or less interchangeably through the literature. As a consequence, if only corners are to be detected it is necessary to do a local analysis of detected interest points to determine which of these are real corners. Examples of edge detection that can be used with post-processing to detect corners are the Kirsch operator and the Frei-Chen masking set.

"Corner", "interest point" and "feature" are used interchangeably in literature, confusing the issue. Specifically, there are several blob detectors that can be referred to as "interest point operators", but which are sometimes erroneously referred to as "corner detectors". Moreover, there exists a notion of ridge detection to capture the presence of elongated objects.

Corner detectors are not usually very robust and often require large redundancies introduced to prevent the effect of individual errors from dominating the recognition task.

One determination of the quality of a corner detector is its ability to detect the same corner in multiple similar images, under conditions of different lighting, translation, rotation and other transforms.

2. Algorithms for corner and edge extraction:

- The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a *computational theory of edge detection* explaining why the technique works.
- The **Moravec corner detector** is one of the earliest corner detection algorithms and defines a *corner* to be a point with low self-similarity. The algorithm tests each pixel in the image to see if a corner is present, by considering how similar a patch centered on the pixel is to nearby, largely overlapping patches. The similarity is

measured by taking the sum of squared differences (SSD) between the corresponding pixels of two patches. A lower number indicates more similarity.

More algorithms will be explained in the next part of the instruction.

2. Harris Detector

The Algorithm:

- Find points with large corner response function R (R > threshold)
- Take the points of local maxima of R

Mathematical theory:

Harris and Stephens improved upon Moravec's corner detector by considering the differential of the corner score with respect to direction directly, instead of using shifted patches. (This corner score is often referred to as autocorrelation, since the term is used in the paper in which this detector is described. However, the mathematics in the paper clearly indicate that the sum of squared differences is used.)

Without loss of generality, we will assume a grayscale 2-dimensional image is used. Let this image be given by I. Consider taking an image patch over the area (u, v) and shifting it by (x, y). The weighted *sum of squared differences* (SSD) between these two patches, denoted S, is given by:

$$S(x,y) = \sum_u \sum_v w(u,v)\ \left(I(u+x, v+y) - I(u,v)\right)^2$$

$$S(x,y) = \sum_u^{} \sum_v^{} w(u,v)\ \left(I(u+x, v+y) - I(u,v)\right)^2$$

$I(u+x, v+y)$ can be approximated by a Taylor expansion. Let I_x and I_y be the partial derivatives of I, such that

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

This produces the approximation

$$S(x,y) \approx \sum_u \sum_v w(u,v)\ \left(I_x(u,v)x + I_y(u,v)y\right)^2,$$

which can be written in matrix form:

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix},$$

where *A* is the structure tensor,

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

This matrix is a Harris matrix, and angle brackets denote averaging (i.e. summation over (u,v)). If a circular window $w(u,v)$ (or circularly weighted window, such as a Gaussian) is used, then the response will be isotropic.

A corner (or in general an interest point) is characterized by a large variation of S in all directions of the vector $\begin{pmatrix} x & y \end{pmatrix}$. By analyzing the eigenvalues of A, this characterization can be expressed in the following way: A should have two "large" eigenvalues for an interest point. Based on the magnitudes of the eigenvalues, the following inferences can be made based on this argument:

1. If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then this pixel (x,y) has no features of interest.
2. If $\lambda_1 \approx 0$ and λ_2 has some large positive value, then an edge is found.
3. If λ_1 and λ_2 have large positive values, then a corner is found.

Harris and Stephens note that exact computation of the eigenvalues is computationally expensive, since it requires the computation of a square root, and instead suggest the following function M_c, where κ is a tunable sensitivity parameter:

$$M_c = \lambda_1 \lambda_2 - \kappa \left(\lambda_1 + \lambda_2\right)^2 = \det(A) - \kappa \ \mathrm{trace}^2(A)$$

Therefore, the algorithm does not have to actually compute the eigenvalue decomposition of the matrix A and instead it is sufficient to evaluate the determinant and trace of A to find corners, or rather interest points in general.

3. The SUSAN corner detector

Theory

SUSAN is an acronym standing for *smallest univalue segment assimilating nucleus.* This method is the subject of a 1994 UK patent, which is no longer in force.

For feature detection, SUSAN places a circular mask over the pixel to be tested (the nucleus). The region of the mask is M, and a pixel in this mask is represented by $\vec{m} \in M$. The nucleus is at $\vec{m}_0$. Every pixel is compared to the nucleus using the comparison function:

$$c(\vec{m}) = e^{-\left(\frac{I(\vec{m}) - I(\vec{m}_0)}{t}\right)^6}$$

where t determines the radius, I is the brightness of the pixel and the power of the exponent has been determined empirically. This function has the appearance of a smoothed top-hat or rectangular function. The area of the SUSAN is given by:

$$n(M) = \sum_{\vec{m} \in M} c(\vec{m})$$

If c is the rectangular function, then n is the number of pixels in the mask which are within t of the nucleus. The response of the SUSAN operator is given by:

$$R(M) = \begin{cases} g - n(M) & \text{if } n(M) < g \\ 0 & \text{otherwise,} \end{cases}$$

where g is named the `geometric threshold'. In other words the SUSAN operator only has a positive score if the area is small enough. The smallest SUSAN locally can be found using non-maximal suppression, and this is the complete SUSAN operator.

The value t determines how similar points have to be to the nucleus before they are considered to be part of the univalue segment. The value of g determines the minimum size of the univalue segment. If g is large enough, then this becomes an edge detector.

For corner detection, two further steps are used. Firstly, the centroid of the SUSAN is found. A proper corner will have the centroid far from the nucleus. The second step insists that all points on the line from the nucleus through the centroid out to the edge of the mask are in the SUSAN.

4. Practice

1. Get your own image file. Create a new script and put image file in the same folder where you have your .m file. Then copy the code below and read it carefully. There are some mistakes to catch. Correct them all.

```
clear all; clc;

% to declare variables
sigma = 1;
radius = 1;
threshold = 1000;
size = 2 * radius + 1;            % the size of matrix filter
dx = [-1 0 1; -1 0 1; -1 0 1];    % x derivative matrix
dy = dx;                          % y derivative matrix
g = fspecial('gaussian', max(1, fix(6*sigma)), sigma);

% To read image
source = imread();

% To get all reds
im = source(:,:,1);

% Step 1: Compute derivatives of image
Ix = conv2(im, dx, 'same');
Iy = conv2(im, dy, 'same');

% Step 2: Smooth spare image derivative
Ix2 = conv2(Ix.^2, g, 'same');
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');


% Step 3: Harris corner measure
harris = (Ix2.*Iy2 - Ixy.^2)./(Ix2+Iy2 + eps);

% Step 4: Find local maximal
mx = ordfilt2(harris, size^2, ones(size));

harris = (harris = mx) & (harris > threshold);

% plot
[rows, cols] = find(harris);
figure, image(source), axis image, colormap(gray), hold on,
plot(cols, rows, 'y'), title('corners detected');
```

2. Do the same thing with the second algorithm.

```
function [ map r c ] = susanCorner( img )

maskSz = [7 7];
fun = @(img) susanFun(img);
map = nlfilter(img,maskSz,fun);
[r c] = find(map);

end

function res = susanFun(img)
% SUSANFUN  Determine if the center of the image patch IMG
%   is corner(res = 1) or not(res = 0)


mask = [...
    0 0 1 1 1 0 0
    0 1 1 1 1 1 0
    1 1 1 1 1 1 1
    1 1 1 1 1 1 1
    1 1 1 1 1 1 1
    0 1 1 1 1 1 0
    0 0 1 1 1 0 0];

% uses 2 thresholds to distinguish corners from edges
thGeo = (nnz(mask)-1)*.2;
thGeo1 = (nnz(mask)-1)*.4;
thGeo2 = (nnz(mask)-1)*.4;
thT = .07;
thT1 = .04;

sz = size(img);
usan = ones(sz)*img(round(sz/2),round(sz/2));

similar = (abs(usan-img)>thT);
similar = similar.*mask;
res = sum(similar(:));
if res < thGeo
    dark = nnz((img-usan<-thT1).*mask);
    bright = nnz((img-usan>thT1).*mask);
    res = min(dark,bright)<thGeo1 && max(dark,bright)>thGeo2;

else
    res = 10;
end

end
```

3. Here are solutions:

I.

1. `dy = dx';`
2. `harris = (harris == mx) & (harris > threshold);`
3. `plot(cols, rows, 'yx'), title('corners detected');`
4. `mx = ordfilt2(harris, size.^2, ones(size));`

II.

1. `sz = size(img);`
2. `similar = (abs(usan-img)<thT);`
3. `res = 0;`