Visual Perception

Practical Lab:
Corner Detection

Related to Lecture 3
Workload: 4 hours in the lab
Programming platform: Matlab

1. Objective

Experience on how to detect corners in an image.
Understand the importance of the different steps involved in corner detection using the Harris algorithm.

2. Corner detection

Introduction

In this practical session we are going to implement the well-known **Harris corner detector**.

This method is based in the use of the image gradient. The procedure is as follows:

Compute the derivatives of the intensity image in the x and y directions for every pixel.

In this way we obtain $I_x = \frac{\partial I}{\partial x}$ $I_y = \frac{\partial I}{\partial y}$ for every point of the image.

Then, we compute the products:

I_x^2, I_y^2 and $I_x I_y$

Next, the squared image derivatives should be smoothed by convolving them with a Gaussian filter g.,

$$\left\langle I_x^2 \right\rangle = g \otimes I_x^2 \tag{1}$$

where $\otimes$ is the convolution operator, and the autocorrelation matrix M can be defined for every pixel:

$$\mathbf{M} = \begin{pmatrix} \sum_W \left\langle I_x^2 \right\rangle & \sum_W \left\langle I_x I_y \right\rangle \\ \sum_W \left\langle I_x I_y \right\rangle & \sum_W \left\langle I_y^2 \right\rangle \end{pmatrix} \tag{2}$$

where *W* is a 3×3 neighborhood around the point.

Remember that M can be used to derive a measure of "cornerness" for every pixel. Depending of the rank of M, the pixel belongs to an homogeneous region (rank M = 0), an edge (significant gradient in 1 direction, that is, rank M = 1), or a corner (significant gradients in both directions, thus, rank M = 2).

Harris and Stephens proposed the use of the determinant and the trace of M to detect corners:

$$R = \det(M) - k \cdot tr(M)^2 \tag{3}$$

where: k is a constant (with a value of 0.04)
$\det(\cdot)$ is the determinant and
tr is the trace (sum of the diagonal elements).

Activity

Go to the course web page and download the files **`corner_detector.m`** and the image **`chessboard00.png`**. The matlab script is just the starting point from which you should carry out this practical session.

```
im = imread('chessboard00.png');

im_orig = im;

if size(im,3)>1 im=rgb2gray(im); end

% Derivative masks

dx = [-1 0 1;
      -1 0 1;
      -1 0 1];
dy = dx';

% Image derivatives
Ix = conv2(double(im), dx, 'same');
Iy = conv2(double(im), dy, 'same');

sigma=2;

% Generate Gaussian filter of size 9x9 and std. dev. sigma.
g = fspecial('gaussian',9, sigma);

% Smoothed squared image derivatives
Ix2 = conv2(Ix.^2, g, 'same');
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');
```

Part 1

Modify the code above to compute a matrix E which contains for every point the value of the smaller eigenvalue of M. Once computed, display this matrix by means of:

```
imshow(mat2gray(E))
```

Part 2

Question

Modify the code above to compute a matrix R which contains for every point the result of Equation (3). What is the difference with respect to E?
Use functions **tic** and **toc** to measure the time required for computing E and R.

Part 3

Question

Select for E and R the 81 most salient points. Is this the result you expected?

Hint for part 3: Build a structure named **`features`** with 2 fields to store the x coordinate (**`p_x`**) and the y coordinate (**`p_y`**) of the 81 points with the highest

the cornerness value given by E or R. Then, display the corners by means of:

```
figure; imshow(im_orig); hold on;

for i=1:size(features,2),
    plot(features(i).p_x, features(i).p_y, 'r+');
end
```

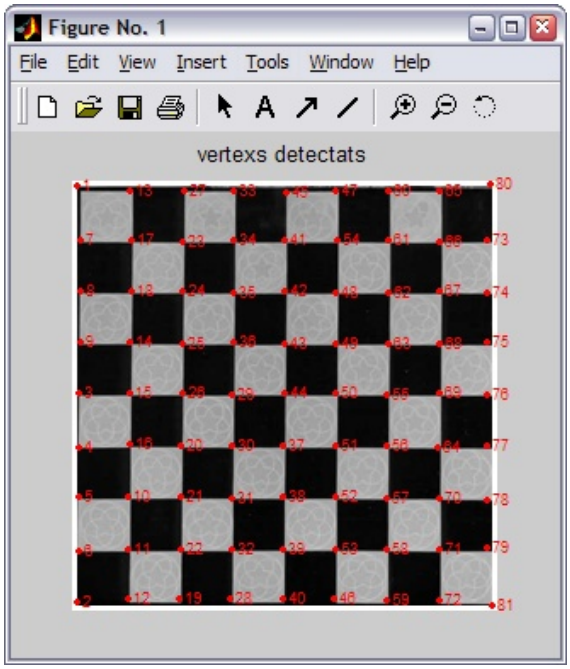
Test your code with images **chessboard01.png**, **chessboard02.png**, etc.

Part 4

Question

Build a function to carry out non-maximal suppression for E and R. Again, the 81 most salient points using a non-maximal suppression of 11×11 pixels. Is this result better than that of Part 3?

Test your code with images **chessboard01.png**, **chessboard02.png**, etc.



Part 5 (this is an optional extrapart)

Question

Find the location of the corners with subpixel accuray. What function would you fit on top of the cornerness values?