Subject: Locating Points of Interest and Exporting Features to Images

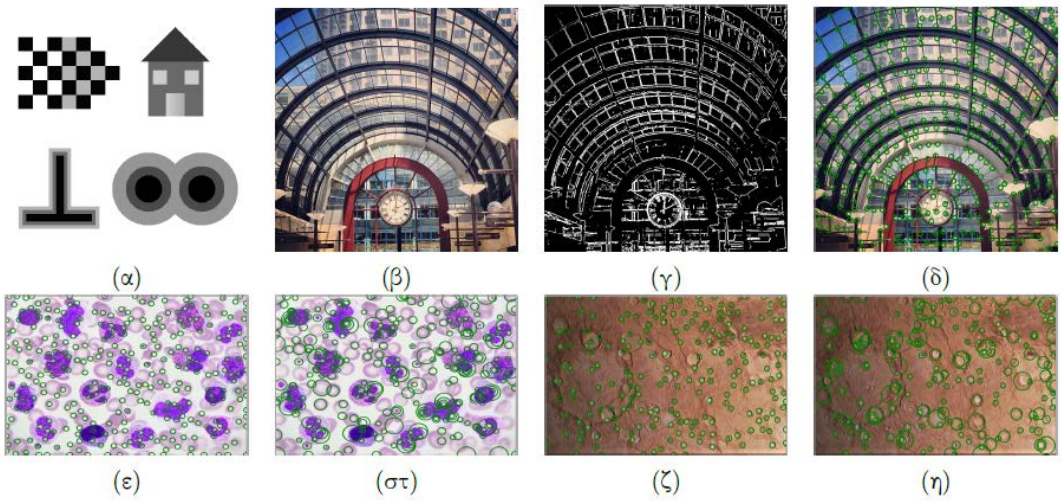

Figure 1: (α) Synthetic image for edge detection, (β) Real image for edge detection, (γ) Edge detection example, (δ) Angle detection example, (ε) Blobs detection example, (στ) Multi-step blob detection example (ζ) Example of blobs detection, (η) Example of multistage blobs detection.

Part 1: Edge Tracking in Gray Images

1.1. Creation of Input Files

1.1.1

Read image I_0 = "edgetest_21.png".

1.1.2

Add noise to I0 to create noisy images $I(x, y) = I_0(x, y) + n(x, y)$ to be used as inputs to the edge detection algorithm. The noise $n(x, y)$ is white gaussian, with an average value of 0 and a standard deviation σ_n such, that the PSNR (Peak-to-peak Signal to Noise Ratio) has specific values. Use 2 different values for the PSNR of the noise:

i) PSNR=20 dB, ii) PSNR=10 dB.

1.2. Implementation of Edge-Detecting Algorithms

Construct an EdgeDetect function that implements the edge detection algorithm described in the following steps:

1.2.1

Generation of impact responses of two distinct linear filters that approach the continuous filters with the following responses:

i) Two-dimensional Gaussian $G_\sigma(x, y)$,

ii) Laplacian-of-Gaussian (LoG) $h(x, y) = \nabla^2 G_\sigma(x, y)$

1.2.2

Approach the Laplacian L of the smoothed image $I_\sigma = G_\sigma * I(x, y)$. Use the following two methods:

i) Linear convolution of I with LoG:

$$L_1 = \nabla^2(G_\sigma * I) = (\nabla^2 G_\sigma) * I = h * I$$

ii) Non-linear (L2): non-linear estimation of Laplacian I with morphological filters

$$L_2 = I_\sigma \oplus B + I_\sigma \ominus B - 2I_\sigma, \quad B = \begin{array}{|c|c|c|}\hline & \bullet & \\ \hline \bullet & \bullet & \bullet \\ \hline & \bullet & \\ \hline \end{array}.$$

1.2.3

Approach of the zerocrossings (points where the entity is zero) of L. A possible algorithm for that is creating the Binary Signum Image (X) of L

```
X = (L >= 0)
```

and finding the contour of X(symmetric with respect to X and X^c)

$$Y = (X \oplus B) - (X \ominus B) \approx \partial X.$$

The zerocrossings correspond to the points at which the binary image Y has the value 1.

1.2.4

Rejection of zerocrossings in relatively smooth areas. Finally, the zerocrossings Y are selected in which the smoothed image I_σ has a relatively large slope, ie the pixels (i, j) for which:

$$Y[i, j] = 1 \quad \kappa\alpha\iota \quad \|\nabla I_\sigma[i, j]\| > \theta_{edge} \cdot \max_{x,y} \|\nabla I_\sigma\|$$

1.3 Evaluation of Edge Detection Results

1.3.1

Calculate the "true" edges using the original clear I_0. The true T edge binary image can be calculated computationally by applying the simple edge operator to I_0:

$$M = (I_0 \oplus B) - (I_0 \ominus B)$$

and using a threshold to convert output to binary image

$$T = M > \theta_{realedge}$$

1.3.2

Quantitative evaluation of edge detection results. Using the binary images of the true T-edges and D-edges detected by your algorithm in the noisy input image, calculate the following quality criterion of the detection result:

$$C = [Pr(D|T) + Pr(T|D)]/2$$

where P r (D | T) is the percentage of the detected edges that are true (Precision) and P r (T | D) the percentage of true edges detected (Recall). If D and T are seen as discrete sets (with pixel elements in which the binary image has a value of 1), then the following holds: P r (T | D) = card (D ∩ T) / card (T), where the card (.) gives the number of elements of a set.

1.3.3

Comment on the effect of changing the parameters and the noise level on the edge detection result. Finally, compare the results for Laplacian's different approaches. In all cases, your commenting should be as concise as possible.

1.4. Application of Edge Detection Algorithms in Real Images

Find the best result and comment on it.

Part 2: Interest Point Detection

2.1 Corner Detection

Use the Harris-Stephens method.

2.1.1

Calculate J1, J2, J3 as:

$$J_1(x,y) = G_\rho * \left(\frac{\partial I_\sigma}{\partial x} \cdot \frac{\partial I_\sigma}{\partial x}\right)(x,y)$$
$$J_2(x,y) = G_\rho * \left(\frac{\partial I_\sigma}{\partial x} \cdot \frac{\partial I_\sigma}{\partial y}\right)(x,y)$$
$$J_3(x,y) = G_\rho * \left(\frac{\partial I_\sigma}{\partial y} \cdot \frac{\partial I_\sigma}{\partial y}\right)(x,y)$$

Where $I_\sigma = G_\sigma * I,$ and G_ρ, G_σ two-dimensional Gaussian normalization nuclei with standard deviations σ (differentiation scale) and ρ (integration scale) respectively.

2.1.2

Calculate λ_+ and λ_- and plot them as gray images.

$$\lambda_{\pm}(x,y) = \frac{1}{2}\left(J_1 + J_3 \pm \sqrt{(J_1 - J_3)^2 + 4J_2^2}\right)$$

2.1.3

Cornerness criterion:

$$R(x,y) = \lambda_-\lambda_+ - k \cdot (\lambda_- + \lambda_+)^2$$, k>0

Choose as corners the pixels that satisfy the conditions:

C1: They are maxima of R within square windows surrounded by the size of which depends on the scale σ, and

C2: They correspond to a value of R greater than a percentage of the total maximum of R, i.e. R (x, y)> θcorn · Rmax, where θcorn a suitably selected threshold.

2.2 Multistage Corner Detection

The method you will implement (Harris-Laplacian) consists of two stages, one for the selection of points on each scale and one for the final selection of the points that show a maximum in some metric unchanged in terms of scale.

2.2.1

The first stage consists of the application of the single-scale angle finding algorithm for different scales of integration and differentiation:

$$\sigma_0, \sigma_1, ..., \sigma_{N-1} = s^0\sigma_0, s^1\sigma_0, ..., s^{N-1}\sigma_0$$
$$\rho_0, \rho_1, ..., \rho_{N-1} = s^0\rho_0, s^1\rho_0, ..., s^{N-1}\rho_0$$

2.2.2

The second stage concerns the automatic selection of the characteristic scale for each point detected in the previous step. First, we calculate the normalized Laplacian of Gaussian LoG for the different scales of differentiation of the previous step:

$$|LoG(\mathbf{x}, \sigma_i)| = \sigma_i^2|L_{xx}(\mathbf{x}, \sigma_i) + L_{yy}(\mathbf{x}, \sigma_i)|, \quad i = 0, ..., N-1$$

We then discard the points for which the scale detected does not maximize the LoG metric in a neighborhood of 2 consecutive scales.

2.3 Blob Detection

To find such areas, in accordance with the Harris method angle criterion, some second-order derivatives of the image are used, namely the delimiter of the Hessian table:

$$H(x,y) = \begin{bmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{bmatrix} \quad (14)$$

όπου $L_{xx}(x,y,\sigma) = \frac{\partial^2}{\partial x^2}\{I_\sigma(x,y)\}$, $L_{yy}(x,y,\sigma) = \frac{\partial^2}{\partial y^2}\{I_\sigma(x,y)\}$ και $L_{xy}(x,y,\sigma) = \frac{\partial^2}{\partial x \partial y}\{I_\sigma(x,y)\}$.

2.3.1

Calculate the some second order derivatives of the image Lxx, Lxy, Lyy by selecting a value for the scale σ and construct the criterion R (x, y) = det (H (x, y)) for each pixel.

2.3.2

Select points of interest that are locally maximal and have a value greater than a properly defined threshold, just as in the Harris method of angle detection.

2.4 Multistage Blob Detection

2.4.1

Repeat the multistage angle detection process, adding a second step to selecting points of interest that are at maximum scale (Hessian-Laplace). You will select the starting points of interest for each scale according to the method of the previous query.

2.5. Acceleration using Box Filters and Integral Images

Hessian's calculation for each scale corresponds to the convergence of the image with increasingly sized filters, which is a computationally expensive process. To speed up this method, [1] (Speeded Up Robust Features) proposed the approach of 2nd derivative filters with Box Filters, ie filters based on sums of rectangular areas, as shown in Figure 2. The implementation of such sums is done very effectively using Integral Images (see Appendix).

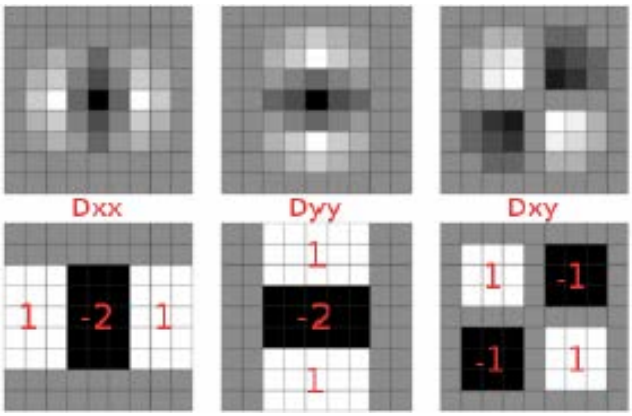

Figure 2: Visualization of the 2nd derivative filter approach with Box Filters

2.5.1

Calculate the integral image

2.5.2

Calculation of Lxx, Lxy, Lyy using the Integral Image according to the Box Filters (Dxx, Dxy, Dyy) for the selected p. At this stage the filters of Figure 2 will be implemented, where each sum of the rectangular filter windows is weighted with the displayed factor. For given σ, which corresponds to a filter of size n × n (with n = 2ceil (3σ) + 1), the size of the windows is calculated as follows:

- D_{xx}: ύψος παραθύρου $4 \times floor(n/6) + 1$, πλάτος παραθύρου $2 \times floor(n/6) + 1$
- D_{yy}: ύψος παραθύρου $2 \times floor(n/6) + 1$, πλάτος παραθύρου $4 \times floor(n/6) + 1$
- D_{xy}: ύψος παραθύρου $2 \times floor(n/6) + 1$, πλάτος παραθύρου $2 \times floor(n/6) + 1$

Where "ύψος παραθύρου" is window height and "πλάτος παραθύρου" is window width.

Note that for each filter we want the local sums for a fixed sized window and the final filter results as a weighted combination of judgments for that window.

2.5.3

Find the points of interest as the local maxima of the criterion R (x, y) in the same way as the previous questions.

$$R(x, y) = L_{xx}(x, y)L_{yy}(x, y) - (0.9L_{xy}(x, y))^2$$

Visualize (as an image) the criterion R of the simple Hessian method and the above approximate criterion R for some scale p. Select indicatively 3-4 scales (σ ∈ (2, 10)) and comment on the quality of approach for increasing scales.

2.5.4

Repeat the multistage process for finding points of interest as explained in previous questions.

Part 3: Applications in Matching and Categorizing Images Using Local Descriptors to Points of Interest

The points of interest give an estimate of areas, which contain important features of the image. For this reason from these areas we export local descriptors, which encode a neighborhood (with a radius that depends on the scale) around the points of interest. You will use the following as local descriptors:

- **SURF** (Speed Up Robust Features)
- **HOG** (Histogram of Oriented Gradients)

Both descriptors are input to a neighborhood of a point of interest and are based on the coding of subset information of that neighborhood using the first directional derivative. The

SURF method first calculates the general direction of the neighborhood so that independent descriptors can be rotated.

3.1. Rotating and Scaling Images Matching

In this section we will examine the ability to find rotation and scale using the point detectors of interest you implemented and the above local descriptors. Specifically for this experiment, 3 images have been used, which we have deformed by rotating them (5 rotations: −20◦, −10◦, 0◦, 10◦, 20◦) and changing their size (4 scales: 0.6, 0.8, 1.0, 1.2). Therefore, we have a total of 20 distortions (similarity transformations) and using one of the images as a reference image, we will try to match its local descriptors with those of the other distorted images. This process is called matching. By choosing a set of point mapping - local descriptors, we can estimate the similarity transformation between the images and consequently the rotation and various scales.

3.1.2

For each of the 5 point detectors you implemented (use only the multistage version of the Box Filters method) and for the two local descriptors (10 combinations in total), calculate and comment on the ability to estimate rotation and scale of images for each combination.

3.2. Image Categorization

This section will evaluate the performance and suitability of the various detectors and descriptors in a typical image categorization problem. You will be given a set of images from the Pascal VOC2005 database. Each image belongs to one of the following three classes: car, man, and bicycle. The purpose is to categorize each image into the correct class using the descriptors you will construct as recognition features.

3.2.1

Initially for each image and from each class you should export the characteristics based on which the categorization will be done. Use the FeatureExtraction function given to you to extract the attributes for the entire base. Use the anonymous functions to use the different detectors / descriptors as in the previous section. Experiment only with multistage versions of descriptors.

3.2.2

Then you have to separate the images in the train and test sets as well as create labels that will show the class to which each image will belong. Use the createTrainTest function given to you to make this separation using the attributes you calculated in the previous query as an argument.

3.2.4

The final stage consists of the final categorization of the images based on the BoVW representation. An SVM Support Ve-ctor Machine classifier appropriately adapted for multiple classes is used for categorization. The whole process is implemented with the svm

function, which returns the result of the recognition as well as the overall success rate. Experiment with different detectors / descriptors and comment on how recognition rates change. What combination would you suggest for the problem of image categorization?