

extracted by command `Xhugeo` of `Xvis` Toolbox. The reader can test Flusser and Gupta moments using commands `Xflusser` and `Xgupta`. \square

5.3 Intensity features

These provide information about the intensity of a region. For gray value images, *e.g.*, X-ray images, there is only one intensity channel. The following features are computed using the gray values in the image, where $x(i, j)$ denotes the gray value of pixel (i, j) .

5.3.1 Basic intensity features

In this Section we summarize basic intensity features that can be easily extracted.

Mean gray value

The mean gray value of the region is computed as:

$$G = \frac{1}{A} \sum_{i,j \in \mathfrak{R}} x(i, j) \quad (5.16)$$

where \mathfrak{R} is the set of pixels of the region and A the area. A 3D representation of the gray values of the region and its neighborhood of our example is shown in Fig. 5.1. In this example $G = 121.90$ ($G = 0$ means 100% black and $G = 255$ corresponds to 100% white).

Mean gradient in the boundary

This feature gives information about the change of the gray values in the boundary of the region. It is computed as:

$$C = \frac{1}{L} \sum_{i,j \in \ell} x'(i, j) \quad (5.17)$$

where $x'(i, j)$ means the gradient of the gray value function in pixel (i, j) (see Section 4.4.1) and ℓ the set of pixels that belong to the boundary of the region. The number of pixels of this set corresponds to L , the perimeter of the region. Using a Gaussian gradient operator in our example in Fig. 5.1, we obtain $C = 35.47$.

Mean second derivative

This feature is computed as:

$$D = \frac{1}{A} \sum_{i,j \in \mathfrak{R}} x''(i,j) \quad (5.18)$$

where $x''(i,j)$ denotes the second derivate of the gray value function in pixel (i,j) . The Laplacian-of-Gauss (LoG) operator can be used to calculate the second derivate of the image. If $D > 0$ we have a region that is darker than its neighborhood as shown in Fig. 4.18.

Other basic features

A simple texture feature is the local variance [102]. This is given by:

$$\sigma_g^2 = \frac{1}{4hb + 2h + 2b} \sum_{i=1}^{2h+1} \sum_{j=1}^{2b+1} (g(i,j) - \bar{g})^2 \quad (5.19)$$

where \bar{g} denotes the mean gray value in the zone.

Other basic intensity features such as kurtosis and skewness can be computed as (5.19). All intensity geometric features explained in this Section can be extracted by command `Xbasicint` of `XVIS` Toolbox. An example is shown in Table 5.2, where the basic 6 intensity features are presented for 10 regions of Fig. 5.2: f_1 : Intensity mean. f_2 : Intensity standard deviation. f_3 : Intensity kurtosis. f_4 : Intensity skewness. f_5 : Mean Laplacian. f_6 : Mean boundary gradient.

Table 5.2: Basic intensity features of apples of Fig. 5.2 [→ Example 5.4 📌]

k	f_1	f_2	f_3	f_4	f_5	f_6
1	158.6886	41.5743	2.2734	-0.5948	-0.3466	11.7224
2	151.1139	39.7086	2.1362	-0.4942	-0.3008	10.4349
3	150.1203	39.7638	2.1516	-0.5286	-0.3117	10.0269
4	148.1976	39.3551	2.0933	-0.4574	-0.3213	10.2263
5	136.1336	34.4244	2.1840	-0.4309	-0.3156	9.0255
6	156.5436	43.7055	2.0797	-0.3863	-0.3287	10.3313
7	152.4726	44.2841	1.9045	-0.3172	-0.3046	9.0481
8	132.3472	33.1643	2.5448	-0.5122	-0.2555	8.8948
9	143.1046	38.3128	2.0296	-0.3942	-0.2908	9.0821
10	166.2491	46.3513	2.2428	-0.5072	-0.3114	11.1117



Matlab Example 5.4: In this example, we show how to extract basic intensity features of ten apples as segmented in Fig. 5.2.

Listing 5.4 : Basic intensity features

```
% AppleBasicIntFeatures.m
close all
I = Xloading('N',1,4)'; % X-ray of apples
```

```

I = I(300:1000,100:2000); % input image
J = and(I>60,I<110);
K = imerode(bwfill(J,'holes'),ones(11,11)); % segmentation
[L,n]=bwlabel(K,4); % regions

X = zeros(n,6); % features
E = zeros(size(I)); % edges of the regions
imshow(I);hold on
op.mask = 15; op.show = 0;
for i=1:n
    Ri = imdilate(L==i,ones(11,11));
    E = or(E,bwperim(Ri));
    X(i,:) = Xbasicint(I,Ri,op); % basic int-features
    c = Xcentroid(Ri); % centroid
    text(c(2),c(1),sprintf('%d',i),... % output
         'color','b','fontsize',12)
end
[ii,jj] = find(E==1); plot(jj,ii,'r.') % display edges
X % features

```

The output of this code is shown in Fig. 5.2 and Table 5.2. The basic geometric features are extracted by command `Xbasicint` of `XVIS` Toolbox. \square

5.3.2 Contrast

The contrast gives a measure of the difference in the gray value between region and its neighborhood. The smaller the gray value difference, the smaller the contrast. In this work, region and neighborhood define a zone. The zone is considered as a window of the image:

$$g(i, j) = x(i + i_r, j + j_r) \quad (5.20)$$

for $i = 1, \dots, 2h + 1$ and $j = 1, \dots, 2w + 1$, where h and w are the height and width as expressed in (5.1). The offsets i_r and j_r are defined as $i_r = \bar{i} - h - 1$ y $j_r = \bar{j} - b - 1$, where (\bar{i}, \bar{j}) denotes the center of mass of the region as computed in (5.12).

Contrast is a very important feature in fault detection, as the differences in the gray values are good for distinguishing a region from its neighborhood. The smaller the gray value difference, the smaller the contrast. In order to visualize the contrast we can use a 3D representation with three coordinates (x, y, z) , where (x, y) are used to represent the location of a pixel (i, j) , and z is used for the representation of the gray value. An example is illustrated in Fig. 5.1c that shows the 3D representation of Fig. 5.1a. The reader can observe in this example a high contrast region.

There are many definitions of contrast. A common definition of contrast is given using texture features (as explained in Section 5.3.5). Other simple definitions of contrast are given in [107, 236]:

$$K_1 = \frac{G - G_e}{G_e}, \quad K_2 = \frac{G - G_e}{G + G_e} \quad \text{y} \quad K_3 = \ln(G/G_e), \quad (5.21)$$

where G and G_e denote the mean gray value in the region and in the neighborhood respectively.

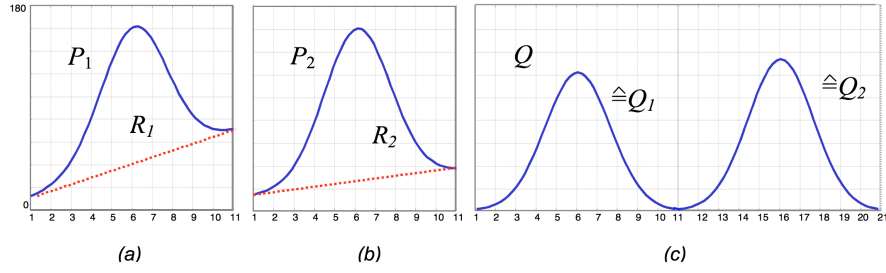


Fig. 5.6: Computation of Q for contrast features for region of Fig. 5.1: a) Profile in i direction, b) profile in j direction, c) fusion of profiles: $Q = [Q_1 \ Q_2]$.

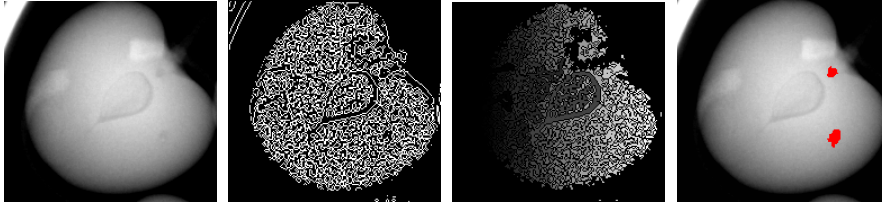


Fig. 5.7: Detection of small defects in apples using area contrast features: input image, edge detection, labeled regions and detection. [→ Example 5.5 📌]

Two further definitions of contrast are given in [167] where new contrast features are suggested. According to Fig. 5.6 these new features can be calculated in four steps: *i*) we take a profile in i direction and in j direction centered in the mass center of the region (see P_1 and P_2 respectively); *ii*) we calculate the ramps R_1 and R_2 that are estimated as a first order function that contains the first and last point of P_1 and P_2 ; *iii*) new profiles without background are computed as $Q_1 = P_1 - R_1$ and $Q_2 = P_2 - R_2$ (they are stored together as $Q = [Q_1 \ Q_2]$); *iv*) the new contrast features are given by:

$$K_\sigma = \sigma_Q \quad \text{and} \quad K = \ln(Q_{\max} - Q_{\min}). \quad (5.22)$$

Another definition of contrast can be found in [111], where the contrast is given by the mean of absolute differences between pixel values and mean of adjacent (*e.g.*, 8-adjacent pixels):

$$K_c = \frac{1}{A_T} \sum_{(i,j) \in \mathbb{T}} |g(i,j) - \mu_{A(i,j)}|. \quad (5.23)$$

where A_T is the area of the region and its neighborhood and $\mu_{A(i,j)}$ is the mean value of pixels locations adjacent of pixel (i, j) .

The contrast features explained in this Section can be computed using command `Xcontrast` of `XVIS` Toolbox.



Matlab Example 5.5: In this example, we show how to detect small defects in an X-ray image of an apple (see Fig. 5.7) using area and contrast features. We follow the general block-diagram of Fig. 4.30. Here, area and contrast features are extracted for each region as defined by enclosed edges. The detection is performed if the size of the region is between some thresholds and the contrast is high enough.

Listing 5.5 : Defects detections using area and contrast features

```
% ContrastDefects.m
close all
I = Xloadimg('N',1,4);           % X-ray of apples
X = I(1450:1629,420:609);       % input image (one apple)
figure(1)
imshow(X); title('input image');

E = and(edge(X,'log',1e-10,1),X>40); % edge detection

[F,m] = bwlabel(not(E),4);       % labels of the regions
op.neighbor = 2;                 % neighborhood is imdilate
op.param = 5;                   % with 5x5 mask
op.show = 0;
R = zeros(size(X));             % initialization of detection
for i=1:m
    Ri = F==i;                  % region i
    Area = sum(Ri(:));
    K = Xcontrast(X,Ri,op);      % contrast features
    if (Area>50) && (Area<150) && ...
        K(2)<-0.01 && K(5) > 2.9 % detection
        R = or(R,Ri);
    end
end
figure(2)
Xbinview(X,imclose(R,ones(5,5))); % output image
title('small defects');
```

The output of this code is shown in Fig. 5.7. In this example, the contrast features are extracted using command `Xcontrast` of `Xvis` Toolbox. In this example we use features K_2 from (5.21) and K from (5.22). □

5.3.3 Crossing line profiles

An approach based on *crossing line profiles* (CLP) was originally developed to detect aluminum casting defects [151], however, it can be used to detect spots in general, or regions that have some gray value difference with their neighborhood. As the contrast between a defect and a defect-free neighborhood is distinctive, the detection is usually performed by thresholding this feature (as we already learned in Section 5.3.2). Nevertheless, this measurement suffers from accuracy error when the neighborhood is not homogeneous, for example when a defect is at an edge of a regular structure of the test object (see Fig. 4.31). For this reason, many approaches use a-priori information about the location of regular structures of the test piece. CLP is able to detect those defects without a-priori knowledge using *crossing line*

profiles, i.e., the gray level profiles along straight lines crossing each segmented potential region in the middle. The profile that contains the most similar gray levels in the extremes is selected. Hence, the homogeneity of the neighborhood is ensured. Features from the selected profile are extracted.

In this approach, we follow a simple automated segmentation approach based on Fig. 4.30 and Fig. 4.31. The steps of detection based on CLP are shown in Fig. 5.8. First, a Laplacian-of-Gaussian (LoG) kernel and a zero crossing algorithm are used to detect the edges of the X-ray images. The LoG-operator involves a Gaussian low-pass filter which is a good choice for the pre-smoothing of our noisy images that are obtained without frame averaging. The resulting binary edge image should produce at real defects closed and connected contours which demarcate *regions*. However, a region of interest may not be perfectly enclosed if it is located at an edge of a regular structure as shown in Fig. 5.8c. In order to complete the remaining edges of these defects, a thickening of the edges of the regular structure is performed as follows: a) the gradient of the original image is calculated (see Fig. 5.8d); b) by thresholding the gradient image at a high gray level a new binary image is obtained; and c) the resulting image is added to the zero-crossing image (see Fig. 5.8e). Afterwards, each

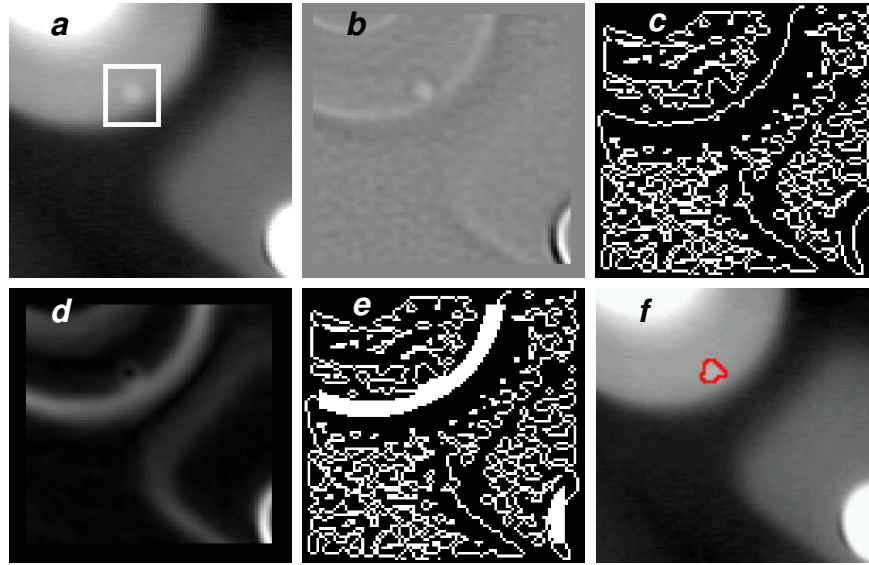


Fig. 5.8: Detection of flaws: a) radioscopic image with a small flaw at an edge of a regular structure, b) Laplacian-filtered image with $\sigma = 1.25$ pixels (kernel size = 11×11), c) zero crossing image, d) gradient image, e) edge detection after adding high gradient pixels, and f) detected flaw using feature F_1 extracted from a crossing line profile. [\rightarrow Example 5.6 \blacktriangle]

closed region is segmented as a potential flaw. For details see a description of the method in [165].

This is a very simple detector of potential regions with a large number of false detections flagged erroneously. However, the advantages are as follows: *i*) it is a single detector (it is the same detector for each image), *ii*) it is able to identify potential defects independent of the placement and the structure of the specimen, *i.e.*, without a-priori information of the design structure of the test piece, and *iii*) the detection rate of real flaws is very high (approximately 90%). In order to reduce the number of the false positives, the segmented regions must be measured and classified.

A segmented potential region is defined as a region enclosed by edges of the binary image obtained in the edge detection (see connected black pixels in Fig. 5.8e). For each segmented region, a window g is defined from the X-ray image x as: $g(i, j) = x(i + i_r, j + j_r)$ for $i = 1 \dots 2h + 1$, and $j = 1 \dots 2w + 1$, where h and w are the height and width of the region as defined in (5.1). The offsets i_r and j_r are defined as $i_r = \bar{i} - h - 1$ and $j_r = \bar{j} - w - 1$ where (\bar{i}, \bar{j}) denotes the coordinates of the center of mass of the region (5.12), rounded to the nearest integers. Hence, g is a window of size $(2h + 1) \times (2w + 1)$, in which the middle pixel corresponds to the center of mass of the segmented potential flaw, *i.e.*, $g(h + 1, w + 1) = x(\bar{i}, \bar{j})$.

Now, we define the crossing line profile P_θ as the gray level function along a straight line of window g through the middle pixel $(h + 1, w + 1)$ forming an angle θ with i -axis. In Section 5.3.2, P_0 and $P_{\pi/2}$ were analyzed together in order to obtain two features, K and K_σ , that give a measurement of the difference between maximum and minimum, and the standard deviation of both crossing line profiles. However, the analysis does not take into account that the profiles could include a non-homogeneous area. For example, if a non-defect region is segmented at an edge of a regular structure, it could be that P_0 (or $P_{\pi/2}$) includes a significant gray level change of the regular structure. In this case, the variation of the profile will be large and therefore the region will be erroneously classified as defect.

In order to avoid this problem, we suggest an individual analysis of eight crossing line profiles P_θ , at $\theta = k\pi/8$, for $k = 0, \dots, 7$, as illustrated in Fig. 5.9. In this analysis, the crossing line profile that contains the most similar gray levels in the extremes is selected. Hence, the attempt is made to ensure the homogeneity of the neighborhood filtering out those profiles that present a high gray level change in the edge of the regular structure. In the example of Fig. 5.9, the selected profile is obtained for $k = 5$ where the gray values of the extremes are both approximately equal to 150. We can observe that the selected crossing line is approximately perpendicular to the direction of the gradient of the X-ray image without defect. This coincides with one of the criteria used by approaches with a-priori knowledge: the selected pixels of the defect-free area are located perpendicular to the direction of the gradient of the piece's contour [168].

Before the features are extracted, a pre-processing of the selected crossing line profile is performed as follows: 1) The selected profile is resized to size $n = 32$ using a nearest neighbor interpolation. The resized profile will be denoted by P . 2) In order to obtain a defect profile without the background of the regular structure, P

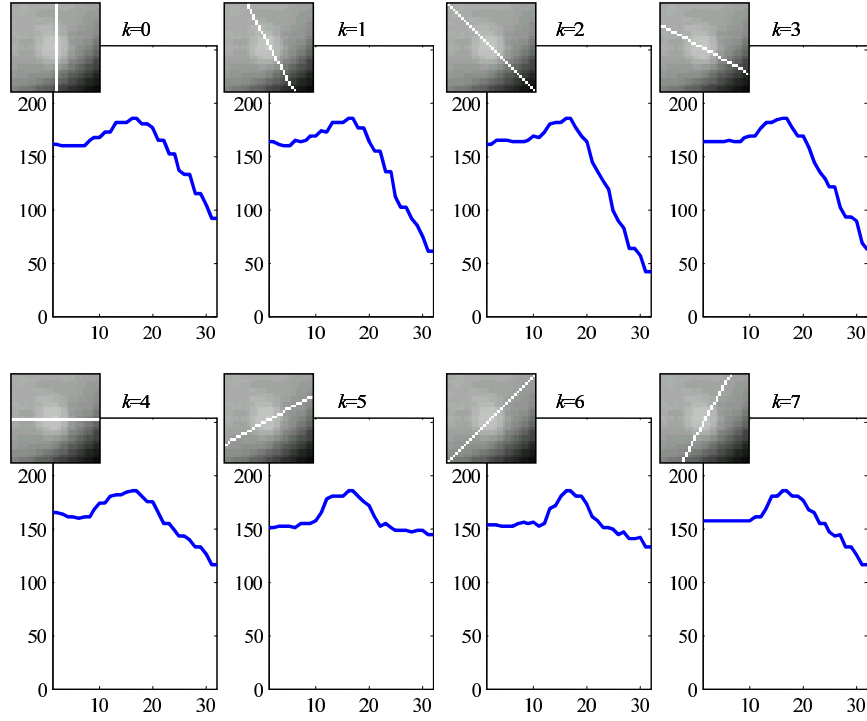



Fig. 5.9: Crossing line profiles for the window shown in Fig. 5.8a. [→ Example 5.6
]

is linearly transformed by $Q_i = mP_i + b$, for $i = 1, \dots, n$, where m and b are so chosen that $Q_1 = Q_n = 0$.

Finally, the proposed features are extracted from the normalized profile Q . They are defined as follows:

$$\begin{aligned}
 \bar{Q} &= \text{mean}(Q) \\
 \sigma_Q &= \text{std}(Q) \\
 \Delta_Q &= \max(Q) - \min(Q) \\
 F_i &= \sum_{k=0}^{n-1} Q_{k+1} e^{-j \frac{2\pi k i}{n}} \text{ for } i = 1, \dots, 4.
 \end{aligned} \tag{5.24}$$

That is \bar{Q} : mean of Q ; σ_Q : standard deviation of Q ; Δ_Q : difference between maximum and minimum of Q ; and F_i : magnitude of the i -th harmonic of the Discrete Fourier Transform of Q for $i = 1, \dots, 4$.



Matlab Example 5.6: In this example, we show how to detect a very small casting defect that is located at the edge of a regular structure as illustrated in Fig. 5.8

using area and CLP features. We follow the general block-diagram of Fig. 4.30. That is area and contrast features are extracted for each region defined by enclosed edges. The detection is performed if the size of the region is between some thresholds and a CLP feature is high enough.

Listing 5.6 : Defects detections using area and CLP features

```
% DetectionCLP.m
close all
X = imread('small_wheel.png');
[N,M] = size(X);
figure(1);
imshow(X); title('Input image');           % input image

D = Xgradlog(X,1.25,4);                     % edge detection
figure(2)
imshow(D,[]);title('or(Log,High gradient)')

[F,m] = bwlabel(not(D),4);                  % labels of the regions
op.ng = 32;                                % size of CLP window
op.show = 0;                               % do not display results
R = zeros(N,M);                            % initialization of detection
for i=1:m                                   % for each region
    Ri = F==i;                             % region i
    Area = sum(Ri(:));                     % area of region
    CLP = Xclp(X,Ri,op);                   % CLP features
    if (Area>10) && (Area<40) && CLP(6)>0.8 % detection
        R = or(R,Ri);
        op.show = 1;
        CLP = Xclp(X,Ri,op);               % display results
        op.show = 0;
        pause(1)
    end
end
figure(3)
Xbinview(X,imclose(R,ones(5,5)));          % output image
title('Casting defects');
```

The output of this code is shown in Fig. 5.8 and 5.9. In this example, the edges are detected using command `Xgradlog` of `ℳvis` Toolbox, that computes the logical OR of edge detection using LoG and edge detection by thresholding the gradient. The contrast features are extracted using command `Xclp` of `ℳvis` Toolbox. In this example we use features F_1 from (5.24). □

CLP features were tested on detecting casting defects. In this experiment, 50 X-ray images of aluminum wheels were analyzed. In the segmentation, approximately 23,000 potential flaws were obtained, in which there were 60 real defects. Some of these were existing blow holes. The other defects were produced by drilling small holes in positions of the casting which were known to be difficult to detect. In the performance analysis, the best result was achieved by our feature F_1 (5.24). The class distribution between class ‘defect’ and ‘non-defect’ (or regular structure) is illustrated in Fig. 5.10. The reader can observe the effectiveness of the separation clearly. For more details see [151].

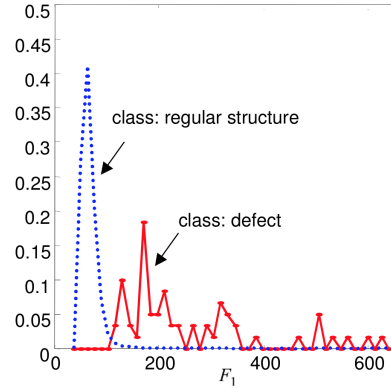


Fig. 5.10: Class distribution of CLP feature F_1 in detection of casting defects.

5.3.4 Intensity moments

In intensity moments, we use statistical moments (5.11) including gray value information [236]:

$$m'_{rs} = \sum_{i,j \in \mathfrak{R}} i^r j^s x(i,j) \quad \text{for } r, s \in \mathbb{N}. \quad (5.25)$$

The summation is computed over the pixels (i, j) of the region \mathfrak{R} only. Thus, it is possible to compute Hu, Flusser and Gupta moments, as explained in Section 5.2.4 using the gray value information of the region. Hu moments with intensity information can be computed by [Xhuint](#) of \mathbb{X} vis Toolbox.

5.3.5 Statistical textures

These features provide information about the distribution of the gray values in the image. In this work however, we restrict the computation of the texture features for a zone only defined as region and neighborhood (see equation 5.20).

Statistical texture features can be computed using the co-occurrence matrix \mathbf{P}_{kl} [75]. The element $P_{kl}(i, j)$ of this matrix for a zone is the number of times, divided by N_T , that gray-levels i and j occur in two pixels separated by that distance and direction given by the vector (k, l) , where N_T is the number of pixels pairs contributing to build matrix \mathbf{P}_{kl} . In order to decrease the size $N_x \times N_x$ of the co-occurrence matrix the gray scale is often reduced to 8 gray levels. From the co-occurrence matrix several texture features can be computed. Haralick in [75] proposes (here $p(i, j) := P_{kl}(i, j)$):

Angular second moment:	$f_1 = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} [p(i, j)]^2$
Contrast:	$f_2 = \sum_{n=0}^{N_x-1} n^2 \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \text{ for } i-j = n$
Correlation:	$f_3 = \frac{1}{\sigma_x \sigma_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} [i \cdot j \cdot p(i, j) - \mu_x \mu_y]^2$
Sum of squares:	$f_4 = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} (i-j)^2 p(i, j)$
Inverse difference moment:	$f_5 = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} \frac{p(i, j)}{1+(i-j)^2}$
Sum average:	$f_6 = \sum_{i=2}^{2N_x} i \cdot p_{x+y}(i)$
Sum variance:	$f_7 = \sum_{i=2}^{2N_x} (i - f_8) \cdot p_{x+y}(i)$
Sum entropy:	$f_8 = -\sum_{i=2}^{2N_x} p_{x+y}(i) \cdot \log(p_{x+y}(i))$
Entropy:	$f_9 = -\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \log(p(i, j))$
Difference variance:	$f_{10} = \text{var}(\mathbf{p}_{x-y})$
Difference entropy:	$f_{11} = -\sum_{i=0}^{N_x-1} p_{x-y}(i) \cdot \log(p_{x-y}(i))$
Information measures of correlation 1:	$f_{12} = \frac{f_9 - HXY1}{\max(HX, HY)}$
Information measures of correlation 2:	$f_{13} = \sqrt{1 - \exp(-2(HXY2 - HXY))}$
Maximal correlation coefficient:	$f_{14} = \sqrt{\lambda_2}$

(5.26)

where μ_x , μ_y , σ_x and σ_y are the means and standard deviations of p_x and p_y respectively with

$$\begin{aligned}
 p_x &= \sum_{j=1}^{N_x} p(i, j) \\
 p_y &= \sum_{i=1}^{N_x} p(i, j) \\
 p_{x+y}(k) &= \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \text{ for } k = 2, 3, \dots, 2N_x \\
 p_{x-y}(k) &= \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \text{ for } k = 0, 1, \dots, N_x - 1
 \end{aligned}$$

and

$$\begin{aligned}
 HX &= -\sum_{i=1}^{N_x} p_x(i) \log(p_x(i)) \\
 HY &= -\sum_{j=1}^{N_x} p_y(j) \log(p_y(j)) \\
 HXY1 &= -\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \log(p_x(i) p_y(j)) \\
 HXY2 &= -\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p_x(i) p_y(j) \log(p_x(i) p_y(j))
 \end{aligned}$$

In f_{14} , λ_2 is the second largest eigenvalue of Q defined by

$$Q(i, j) = \sum_{k=1}^{N_x} \frac{p(i, k) p(j, k)}{p_x(i) p_y(k)}$$

The texture features are extracted for four directions (0° - 180° , 45° - 225° , 90° - 270° and 135° - 315°) in different distances $d = \max(k, l)$. That is, for a given distance d we have four possible co-occurrence matrices: P_{0d} , P_{45d} , P_{90d} and P_{135d} . For example, for $d = 1$, we have $(k, l) = (0, 1)$; $(1, 1)$; $(1, 0)$; and $(-1, 1)$. After Haralick, fourteen texture features using each co-occurrence matrix are computed (5.26), and the mean and range for each feature are calculated, *i.e.*, we obtain $14 \times 2 = 28$ tex-

ture features for each distance d . The features will be denoted as \bar{f}_i for the mean and f_i^Δ for the range, for $i = 1 \dots 14$.

The texture features after Haralick can be computed using command `Xharalick` of `ℳVIS` Toolbox.

5.3.6 Gabor

The Gabor functions are Gaussian shaped band-pass filters, with dyadic treatment of the radial spatial frequency range and multiple orientations, which represent an appropriate choice for tasks requiring simultaneous measurement in both space and frequency domains. The Gabor functions are a complete (but a nonorthogonal) basis set given by:

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right) \quad (5.27)$$

where σ_x and σ_y denote the Gaussian envelope along the x and y -axes, and u_0 defines the radial frequency of the Gabor function. Examples of Gabor functions are illustrated in Fig. 5.11. In this case a class of self-similar functions are generated by rotation and dilation of $f(x, y)$.

Each Gabor filter has a real and an imaginary component that are stored in $M \times M$ masks, called \mathbf{R}_{pq} and \mathbf{I}_{pq} respectively, where $p = 1 \dots S$, denotes the scale, and $q = 1 \dots L$, denotes the orientation (for details see [118]). Usually, $S = 8$ scales, and $L = 8$ orientations as shown in Fig. 5.11, with $M = 27$.

The Gabor filters are applied to each segmented window \mathbf{W} , that contains the segmented region and its surrounding (see Fig. 5.1). The filtered windows \mathbf{G}_{pq} are computed using the 2D convolution (4.9) of the window \mathbf{W} of the X-ray image with the Gabor masks as follows:

$$\mathbf{G}_{pq} = [(\mathbf{W} * \mathbf{R}_{pq})^2 + (\mathbf{W} * \mathbf{I}_{pq})^2]^{1/2} \quad (5.28)$$

The Gabor features, denoted by g_{pq} , are defined as the average output of \mathbf{G}_{pq} , *i.e.*, it yields $S \times L$ Gabor features for each segmented window:

$$g_{pq} = \frac{1}{n_w n_w} \sum_{i=1}^{n_w} \sum_{j=1}^{m_w} G_{pq}(i, j) \quad (5.29)$$

where the size of the filtered windows \mathbf{G}_{pq} is $n_w \times m_w$.

Three additional Gabor features can be extracted: *i*) maximum of all Gabor features: $g_{\max} = \max(\mathbf{g})$, *ii*) minimum of all Gabor features: $g_{\min} = \min(\mathbf{g})$, and *iii*) range of all Gabor features: $g_\Delta = g_{\max} - g_{\min}$. These features are very useful because they are rotation invariant.

The Gabor features can be computed using command `Xgabor` of `ℳVIS` Toolbox.

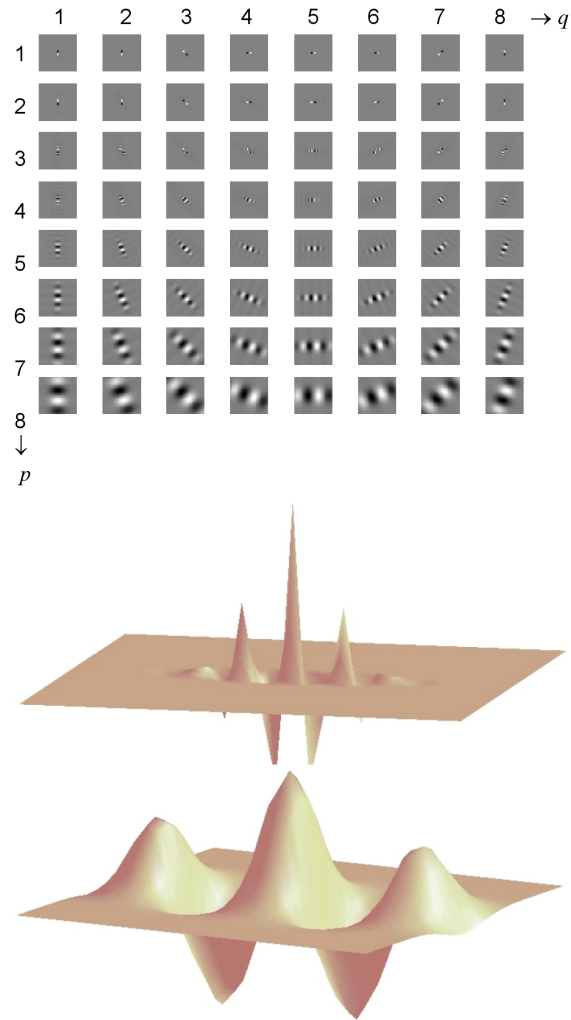


Fig. 5.11: Example of Gabor functions in spatial domain: Top) imaginary components of self-similar filter bank by using $p = 1 \dots 8$ scales and $q = 1 \dots 8$ orientations, Bottom) 3D representations of two Gabor functions of a).

5.3.7 Filter banks

Filter banks can be used to extract texture information [209]. They are used in image transformations like Discrete Fourier Transform (DFT) (magnitude and phase), Discrete Cosine Transform (DCT) [66], and Wavelets as Gabor features based on 2D Gabor functions (see Section 5.3.6).

For an image \mathbf{X} of $N \times N$ pixels, the Discrete Fourier Transformation in 2D is defined as follows:

$$F(m, n) = \sum_{i=1}^M \sum_{k=1}^N X(i, k) e^{-2\pi j \left(\frac{(m-1)(i-1)}{N} + \frac{(n-1)(k-1)}{N} \right)} \quad (5.30)$$

where $j = \sqrt{-1}$. $F(m, n)$ is a complex number. That means magnitude and phase can be used as features. Fourier features can be computed using command `Xfourier` of `ℳvis` Toolbox.

Discrete Cosine Transform in 2D is defined as:

$$D(m, n) = \alpha_m \alpha_n \sum_{i=1}^N \sum_{k=1}^N X(i, k) \cos \left(\frac{\pi(2i-1)(m-1)}{2N} \right) \cos \left(\frac{\pi(2k-1)(n-1)}{2N} \right) \quad (5.31)$$

where $\alpha_1 = 1/\sqrt{N}$ and $\alpha_m = \sqrt{2/N}$, for $m = 2 \dots N$. DCT features are real numbers instead of complex number such as Fourier features. DCT features can be computed using command `Xdct` of `ℳvis` Toolbox.

It is worth mentioning that these features are not rotation invariant, however, we can extract rotation invariant features if we use maximum, minimum and a range of them as we did for the Gabor features in Section 5.3.6.

5.4 Descriptors

Descriptors have been very relevant on computer vision applications [179]. This is because they are able to provide highly distinctive features, and can be used in applications such as multiple view analysis, in object recognition, texture recognition, and others. In this Section we provide some descriptors that are very useful in X-ray testing.

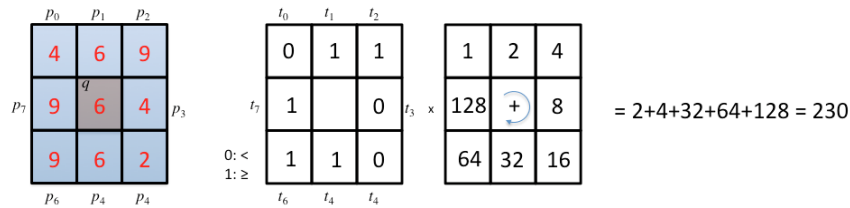


Fig. 5.12: LBP coding: a central pixel q the gray value of which is 6 has 8 neighbors with gray values $p_0 = 4, p_1 = 6, p_2 = 9, p_3 = 4, p_4 = 2, p_5 = 6, p_6 = 9, p_7 = 9$. A new mask with 8 bits is built where $t_i = 1$ if $p_i \geq q$ otherwise $t_i = 0$. The LBP code is computed as $\sum_i t_i 2^i$, in this example the code is 230.

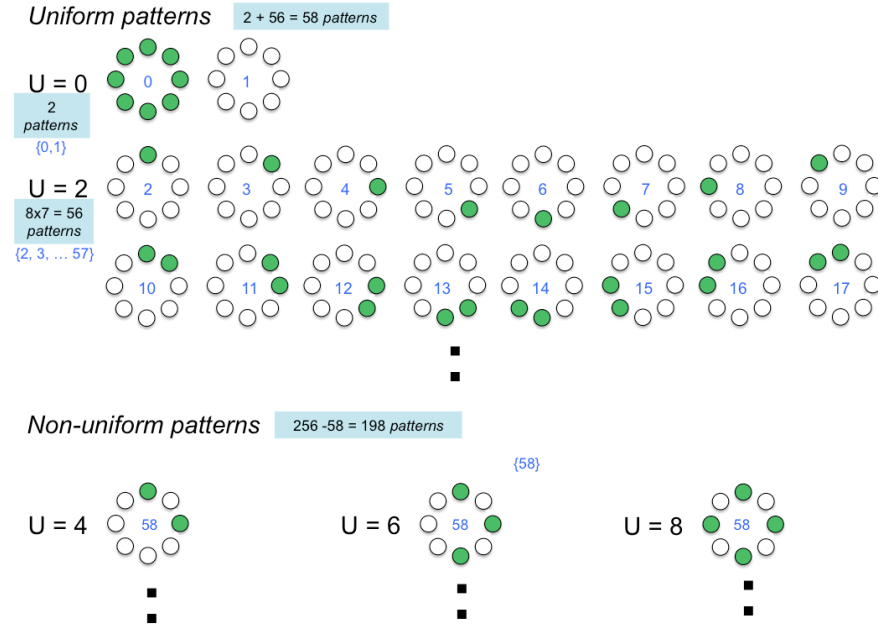


Fig. 5.13: LBP mapping for 8 neighbors. Each small circle represents a bit t_i of the code, green means ‘1’, white means ‘0’. U is the number of changes from ‘1’ to ‘0’, or from ‘0’ to ‘1’ in one cycle. For a small number of changes, *i.e.*, $U = 0$ and 2, the codes represent *uniform patterns*, for $U > 2$ the patterns are *non-uniform*.

5.4.1 Local binary patterns

LBP, *Local Binary Patterns* was proposed as a texture feature [196]. The idea is to extract texture information from occurrence histogram of local binary patterns computed from the relationship between each pixel intensity value with its eight neighbors. The LBP features are the frequencies of each one of the histogram bins. LBP is computed in three steps: *i*) coding, *ii*) mapping and *iii*) histogram.

Coding

Each pixel (i, j) of the input image has a set of neighbors. Typically, the set of eight neighbors defined by the 8-connected pixels is used. However, more neighbors for different distances can be defined as well. For 8 connected pixels, the locations are $(i-1, j-1)$; $(i-1, j)$; $(i-1, j+1)$; $(i, j+1)$; $(i+1, j+1)$; $(i+1, j)$; $(i+1, j-1)$ and $(i, j-1)$ respectively as shown in Fig. 5.12. The central pixel has a gray value q , and the neighbors have gray values p_i , for $i = 0 \dots 7$. The code is computed by:

$$y = \sum_{i=0}^7 t_i 2^i, \quad (5.32)$$

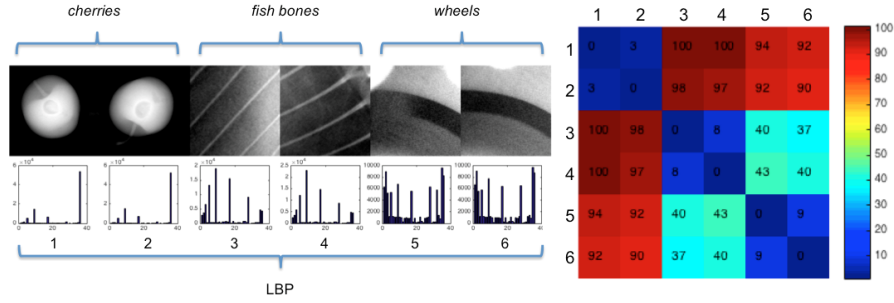


Fig. 5.14: Comparison of six textures using LBP-ri descriptor. It is clear that descriptors of the same texture are very similar, and descriptors from different textures are very different. A measurement of the Euclidean distance between all six descriptors is shown in the right color matrix.

where $t_i = 1$ if $p_i \geq q$ otherwise $t_i = 0$. That means, a pixel q with its neighbors can be coded as a number $y \in \{0 \dots 255\}$. The code can be represented as a string of bits as shown in Fig. 5.12.

Mapping

We can observe that the code generated by the previous step can be categorized according to number of changes (from '1' to '0', or from '0' to '1') in a cycle. For instance, in the example of Fig. 5.12, where the code is 01100111, we define a cycle with eight transitions as: $0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 0$ (the last bit is the repetition of the first one because it is a cycle). The number of changes is $U = 4$. Thus, we can have codes with $U = 0, 2, 4, 6$ and 8 as illustrated in Fig. 5.13. After the authors, there are *uniform* and *non-uniform* patterns. The first ones ($U = 0$ and 2) correspond to textures with a low number of changes, the last ones ($U > 2$) can be interpreted as noise because there are many changes in the gray values. There are 58 uniform patterns and 198 non-uniform patterns. Each uniform code is mapped as a number from 0 to 57 as illustrated in Fig. 5.13, whereas all non-uniform codes are mapped as number 58. This descriptor is known as LBP-u2.

LBP-u2 mapping correspond to a mapping that varies with the orientation of the image, *i.e.*, it is not rotation-invariant. In order to build a rotation-invariant LBP descriptor, all patterns that have the same structure but with different rotations are mapped as an unique number. For instance, all patterns of the second row of Fig. 5.13 are mapped with the same number. The same is valid for the third row. In this mapping, we have 36 different numbers. This descriptor is known as LBP-ri.

Histogram

The process of coding and mapping is performed at each pixel of the input image. Thus, each pixel is converted into a number from 0 to $M - 1$, with a mapping of M

numbers. Afterwards, a histogram of M bins of this image is computed. The LBP descriptor of the image is this histogram.

LBP is very robust in terms of gray-scale and rotation variations [196]. An example is shown in Fig. 5.14. Other LBP features like *semantic* LBP (sLBP) [184] can be used in order to bring together similar bins. LBP is implemented in function `Xlbp` of \mathbb{X} vis Toolbox.

The reader can find a descriptor with similar properties in [197], where LPQ (from *local phase quantization*) is proposed.

5.4.2 Binarized statistical image features

BSIF, *binarized statistical image features*, was proposed as a texture descriptor [108]. As LBP and LPQ, it computes a binary code for each pixel of the input image. Thus, a histogram that encodes texture information is built by counting the frequency of each code.

In BSIF, the input image is filtered using a set of linear filters. The linear filters are learned from a training set of natural image patches ensuring statistical independence of the filter responses. BSIF computes the bits of the binary code by thresholding the response of the linear filters.

Therefore, instead of manually predefined sets of filters (like LBP or LPQ), BSIF uses filters based on statistics of natural images. After the authors, this improves its modeling capacity and the accuracy in texture recognition. BSIF is implemented in function `Xbsif` of \mathbb{X} vis Toolbox. The reader can use this function to obtain similar results to those obtained by LBP in Fig. 5.14.

5.4.3 Histogram of oriented gradients

HOG, *histogram of oriented gradients*, was originally proposed as a descriptor that is able to detect pedestrians [37], however, the powerful of this descriptor can be used in many computer vision problems that require local object appearance and shape information. The key idea of HOG is to compute the distribution of intensity gradients in uniformly spaced cells arranged in a grid manner. A cell is typically defined as a squared region of the image.

In HOG, the gradient of the input image in both directions G_i and G_j is computed (see Section 4.4.1). Thus for each pixel, we have the magnitude $G(i, j)$ and the angle $A(i, j)$ using (4.14) and (4.15) respectively¹. In order to compute the cell histogram, we define n bins, where bin k corresponds to the orientation between θ_k and θ_{k+1} , with $\theta_{k+1} = \theta_k + \Delta\theta$, for $k = 1 \dots n$. For example, for $n = 9$ bins we could define $\Delta\theta = 360^\circ/9 = 40^\circ$ and $\theta_1 = -\Delta\theta/2 = -20^\circ$, so the first bin will be for orienta-

¹ Sometimes the magnitude of the angle is used.

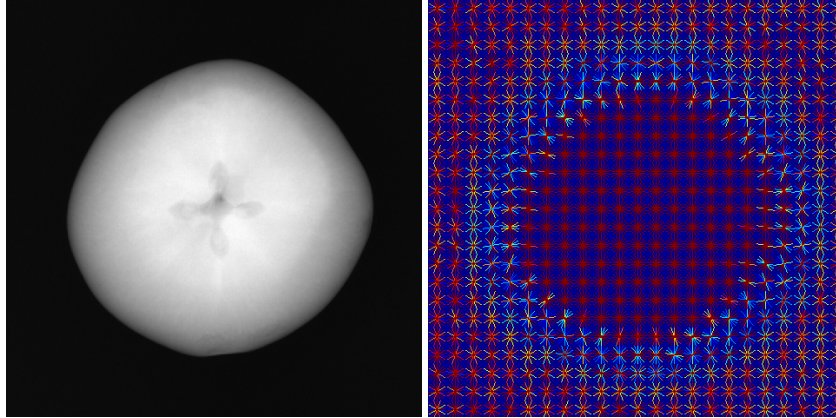


Fig. 5.15: Computation of HOG descriptors of an X-ray image of a fruit. The descriptors give information about shape and appearance.

tions from -20^0 to $+20^0$, the second from $+20^0$ to $+60^0$ and so on. Therefore, a pixel (i, j) of the cell whose orientation is $\theta_k < A(i, j) \leq \theta_{k+1}$, registers a weighted vote in bin k based on its gradient value $G(i, j)$. This operation is repeated for every pixel of the cell.

In order to improve the performance of HOG descriptor and make it robust against changes in illumination and contrast, the authors used a dense grid of cells and an overlapping local contrast normalization [37]. That means, normalized cells are grouped together into connected blocks. Then, the descriptor is a concatenation of the normalized cell histograms of all blocks. HOG is implemented in function `Xhog` of `Xvis` Toolbox. An example is illustrated in Fig. 5.15.

5.4.4 Scale-invariant feature transform

The *Scale-invariant feature transform*, SIFT, was proposed in [130] to detect and describe *keypoints*. A *keypoint* is a distinguishable point in an image, *i.e.*, it represents a salient image region that can be recognized by changing its viewpoint, orientation, scale, etc. In SIFT methodology, each *keypoint* is described using a 128-element vector called *SIFT-descriptor*. SIFT-descriptor is:

- Scale invariant
- Rotation invariant
- Illumination invariant
- Viewpoint invariant

SIFT-descriptor can be used as a ‘signature’ and it is highly distinctive, *i.e.*, SIFT-descriptors of corresponding points (in different images) are very similar, and SIFT-

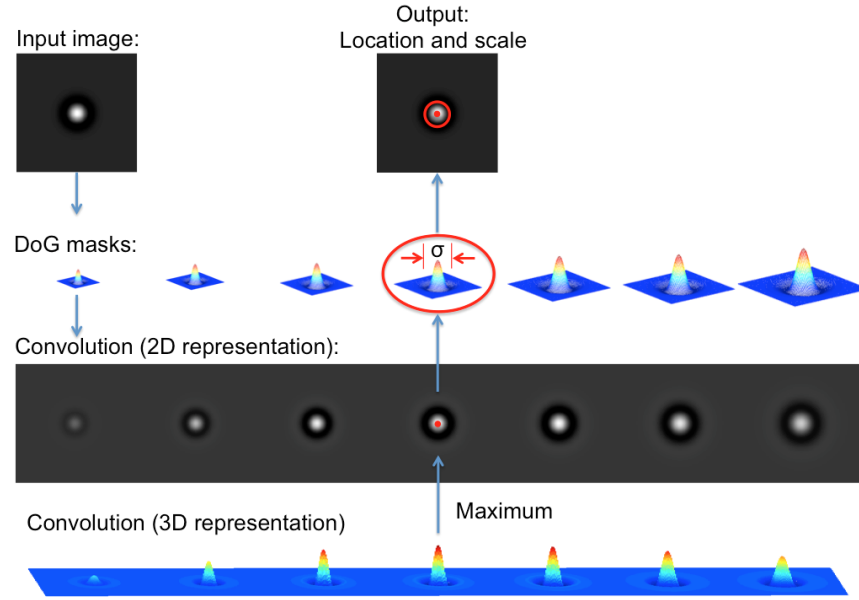


Fig. 5.16: Detection of a keypoint in a synthetic image. The image is convolved with several DoG masks. The maximal response defines the location (x, y) of the keypoint. The used mask for the convolution defines the scale σ .

descriptors of different points are very different. SIFT has two main stages: *i*) keypoint detection and *ii*) keypoint description. In the following, these stages are presented in further details.

Keypoint detection

Keypoints are then taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. Keypoints can be detected in four steps (see Fig. 5.16):

1. We define two Gaussian masks: $G(x, y, \sigma)$ and $G(x, y, k\sigma)$ from (4.10) at scales σ and $k\sigma$.
2. The input image $I(x, y)$ is convolved with both Gaussian filters obtaining $L(x, y, \sigma)$ and $L(x, y, k\sigma)$ respectively.
3. The Difference of Gaussians (DoG) is computed as:

$$D(x, y, \sigma) = L(x, y, \sigma) - L(x, y, k\sigma). \quad (5.33)$$

4. Keypoints are found as maxima of $|D(x, y, \sigma)|$ that can occur at different values of σ . We compare each pixel in the DoG images to its 26 neighbors (8 at the same scale and 9 from the next scale and 9 from the previous scales). If the pixel

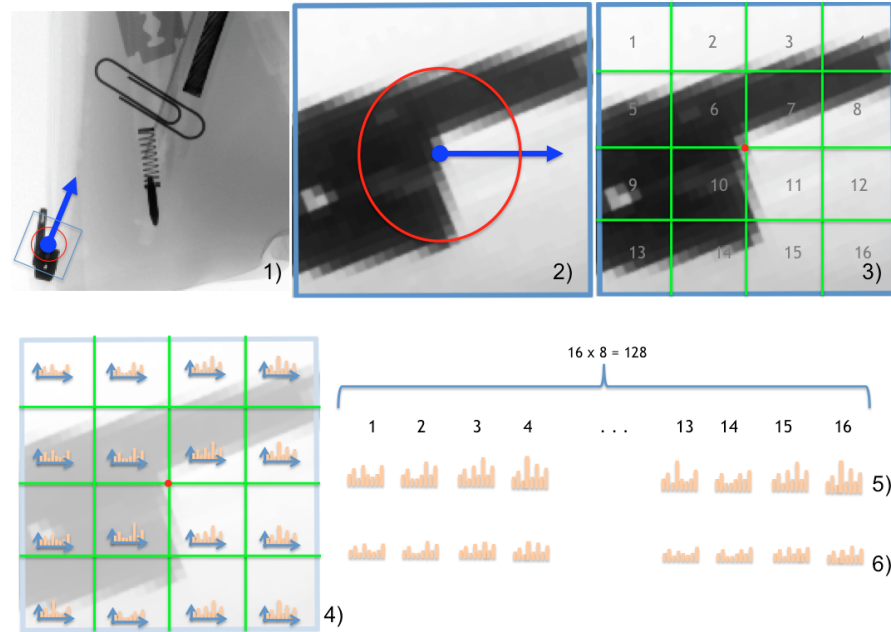


Fig. 5.17: Keypoint description (see explanation of six steps in text). This example corresponds to keypoint number 18 in Fig. 5.18.

value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint.

Keypoint description

For each keypoint we need a description. A keypoint is defined by its location (x, y) and its scale σ . The descriptor is computed in seven steps (see Fig. 5.17):

1. We define a window of size 1.5σ centered in (x, y) .
2. The window is rotated $-\theta$, where θ is the orientation of the gradient in (x, y) .
3. The rotated window is divided into $4 \times 4 = 16$ regular cells distributed in a grid manner.
4. For each cell, the histogram of gradients is computed using 8 bins.
5. All 16 histograms with 8 bins are concatenated, *i.e.*, we obtain a descriptor of $16 \times 8 = 128$ elements.
6. Finally, the descriptor is normalized to unit length.

We can observe that SIFT descriptor is invariant to scale because the size of the window in step 1 depends on scale factor σ . SIFT descriptor is invariant to rotation, because the window is rotated according to the orientation of the gradient (see step 2). Thus, if an image is resized and rotated it will have the same window after these two steps. The SIFT descriptor is invariant to illumination because the descriptor

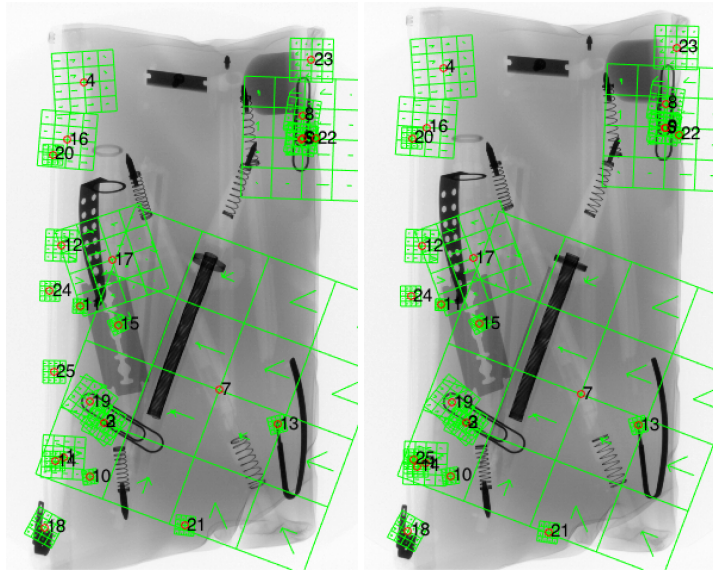


Fig. 5.18: Matching points of two different views of the same object. The object was rotated 10° around its horizontal axis from first to second image. The SIFT approach is able to find key points (red points) and descriptors (represented as green histograms). The descriptors that are similar can be matched. The figure shows the best 25 matching pair points. [→ Example 5.7 ↗]

is normalized to unit length. SIFT has been proven to be robust against perspective distortions and viewpoint changes when the rotation of the 3D object is less than 30 degrees rotation. An example of this can be found in Fig. 5.18.



Matlab Example 5.7: In this example, we find matching points in two views. SIFT keypoints are estimated in each view, and those with the most similar descriptors are matched.

Listing 5.7 : Defects detections using area and CLP features

```
% SIFTmatching.m

I1 = single(Xloading('B',2,1)); % image 1
I2 = single(Xloading('B',2,2)); % image 2
figure(1); imshow(I1,[]); hold on
figure(2); imshow(I2,[]); hold on

[f1,d1] = vl_sift(I1); % SIFT descriptors for image 1
[f2,d2] = vl_sift(I2); % SIFT descriptors for image 2
[mt,sc] = vl_ubcmatch(d1,d2); % matching points
[ii,jj] = sort(sc); % sort of scores
mt = mt(:,jj); sc = sc(:,jj);
n = 25; % the best 25 matchings are selected

figure(1) % display results on image 1
```

```

h1 = vl_plotsiftdescriptor(d1(:,mt(1,1:n)),f1(:,mt(1,1:n))) ;
set(h1,'color','g') ;
for i=1:n
    plot(f1(1,mt(1,i)),f1(2,mt(1,i)),'ro')
    text(f1(1,mt(1,i))+5,f1(2,mt(1,i)),num2str(i),'fontsize',15)
end

figure(2) % display results on image 2
h2 = vl_plotsiftdescriptor(d2(:,mt(2,1:n)),f2(:,mt(2,1:n))) ;
set(h2,'color','g') ;
for i=1:n
    plot(f2(1,mt(2,i)),f2(2,mt(2,i)),'ro')
    text(f2(1,mt(2,i))+5,f2(2,mt(2,i)),num2str(i),'fontsize',15)
end

```

The output of this code is shown in Fig. 5.18. In this example, the SIFT descriptors are detected using command `vl_sift` and matched with command `vl_ubcmatch` of VLfeat Toolbox [253]. □

The reader can find descriptors with similar properties in SURF: *Speeded Up Robust Feature* [15], BRIEF: *Binary robust independent elementary features* [24], BRISK: *Binary Robust Invariant Scalable Keypoints* [122] among others.

5.5 Sparse representations

In recent years, sparse representation has been widely used in signal processing [217], neuroscience [200], statistics [41], sensors [269] and computer vision [265,270]. In many computer vision applications, under assumption that natural images can be represented using sparse decomposition [199] state-of-the-art results have been significantly improved. In these applications, the performance can be improved by learning non-parametric dictionaries for the sparse representation (instead of using fixed dictionaries).

In signal processing, it is very convenient to estimate a new representation of a signal in order to analyze it efficiently. The idea is that this representation captures a useful characterization of the signal for analytical tasks, *e.g.*, feature extraction for pattern recognition, frequency spectrum for denoising, etc. An appropriate representation, due to its simplicity, is obtained by a linear transform. Thus, a signal $\mathbf{x} \in \mathbb{R}^n$ can be expressed as a linear combination of a set of elementary signals $\mathbf{D} = [\mathbf{d}_1 \ \mathbf{d}_2 \ \dots \ \mathbf{d}_K] \in \mathbb{R}^{n \times K}$ as:

$$\mathbf{x} = \mathbf{D}\mathbf{z}, \quad (5.34)$$

where the vector $\mathbf{z} \in \mathbb{R}^K$ corresponds to the representation coefficients of signal \mathbf{x} . In this representation, matrix \mathbf{D} and its columns \mathbf{d}_k are commonly known as *dictionary* and *atoms* respectively.