

# T-61.5070 COMPUTER VISION

## Exercises 2008

### Contents

Exercise 1/08 . . . . .	28
Exercise 2/08 . . . . .	34
Exercise 3/08 . . . . .	41
Exercise 4/08 . . . . .	48
Exercise 5/08 . . . . .	52
Exercise 6/08 . . . . .	61
Exercise 7/08 . . . . .	66
Exercise 8/08 . . . . .	70
Exercise 9/08 . . . . .	75
Exercise 10/08 . . . . .	79
Exercise 11/08 . . . . .	84

**T-61.5070 COMPUTER VISION, Exercise 1/08****Motivation**

The purpose of this exercise is to be acquainted with digital image processing and illumination effects.

1. How can a hexagonal grid be represented in a computer memory in such a way that the image processing is fast? How can the nearest neighbours be found? Write a small program that computes an average in a small neighbourhood (distance=1) and replaces the center point with this average.
2. Consider a line-scan camera. Your intention is to use it in image capturing. You should compensate illumination, optics, and nonhomogenities between the sensors. Derive a formula for this purpose. Both the light and the dark calibration targets are available.
3. Consider the illumination problem with a plane and a sphere. Assume that the surface of the object is perpendicular to the observer and the illumination source. The object is either a plane or a sphere. Assume also that the surfaces are Lambertian ones. The light source is assumed to be a point source.
  - (a) How does the radiance vary on the plane?
  - (b) How does the radiance vary on the sphere?
4. Consider the illumination problem with a plane and two illumination sources. The surface of the plane can be assumed to be a Lambertian one. The object is illuminated by two point sources. The light sources are at the same distance from the plane. Determine the radiance on the plane.

## T-61.5070 COMPUTER VISION, Exercise 1/08

### 1.

The hexagonal grid is depicted in Fig. 1. The coordinates give the memory locations of grid elements. The six nearest neighbors of element (2, 1) in Fig. 1 are

$$N_{(2,1)} \in \{(3,0), (2,0), (1,0), (1,1), (2,2), (3,1)\}.$$

The nearest neighbors of each grid element are

$$N_{(i,j)} = \begin{cases} (i+1, j-1), (i, j-1), (i-1, j-1), (i-1, j), (i, j+1), (i+1, j) & \text{if } i \text{ is even,} \\ (i+1, j), (i, j-1), (i-1, j), (i-1, j+1), (i, j+1), (i+1, j+1) & \text{if } i \text{ is odd.} \end{cases} \quad (1)$$

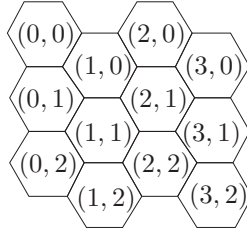


Figure 1: Hexagonal grid in memory.

The average for six nearest neighbors is calculated and substituted to each element, the image boundaries are not considered. This can be done by the following program:

```
for(i=1;i<N-1;i=i+2) /* process odd-i elements */
  for(j=1;j<N-1;j=j+1)
    ave[i][j] = (x[i][j] + x[i+1][j] + x[i][j-1] +
                x[i-1][j] + x[i-1][j+1] + x[i][j+1] +
                x[i+1][j+1])/7;

for(i=2;i<N-1;i=i+2) /* process even-i elements */
  for(j=1;j<N-1;j=j+1)
    ave[i][j] = (x[i][j] + x[i+1][j-1] + x[i][j-1] +
                x[i-1][j-1] + x[i-1][j] + x[i][j+1] +
                x[i+1][j])/7;
```

Fig. 2 depicts distances in a hexagonal grid. Euclidian distances are calculated in a hexagonal grid from displacements  $d_i$  and  $d_j$  in directions  $i$  and  $j$ .

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{d_i^2 + d_j^2}. \quad (2)$$

If  $i_1 - i_2$  is even, i.e. both elements are either in  $C_e \in \{c_0, c_2, c_4, \dots\}$  or in  $C_o \in \{c_1, c_3, c_5, \dots\}$ , the displacement  $d_i$  is

$$d_i = \frac{\sqrt{3}(i_1 - i_2)}{2} \quad (3)$$

and the displacement  $d_j$  is simply

$$d_j = j_1 - j_2. \quad (4)$$

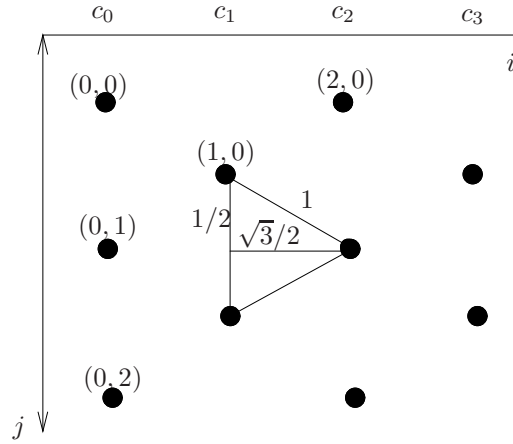


Figure 2: Distances in a hexagonal grid.

When  $i_1 - i_2$  is odd, the displacement  $d_i$  is obtained from Eq. 3 but displacement  $d_j$  from Eq. 4 would be 0.5 units too large for even  $i_1$  ( $i_1 \in C_e$ ) and 0.5 units too small for an odd  $i_1$  ( $i_1 \in C_o$ ). Therefore,  $d_j$  for an even  $i_1$  is

$$d_j = j_1 - j_2 - \frac{1}{2}. \quad (5)$$

and for an odd  $i_1$

$$d_j = j_1 - j_2 + \frac{1}{2}. \quad (6)$$

Euclidean distances may be computed with the following program

```
diff = j1 - j2;
if (((i1 - i2) % 2) != 0) {
    if ((i1%2) == 0)
        diff -= 0.5;
    else
        diff += 0.5;
}
dist = diff * diff;
diff = i1 - i2;
dist += 0.75 * diff * diff;
dist = sqrt(dist);
```

## 2.

Position-dependent brightness correction is represented in Sec. 4.1/5.1, pp. 58–59/114–115. The correction is made, for each image pixel ( $i$ ), with the light and dark calibration targets,  $g_l(i)$  and  $g_d(i)$ . For a linear degradation, the degraded image,  $f(i)$ , is given by

$$f(i) = k(i)g(i) + b(i) \quad (7)$$

The calibration values for  $k(i)$  and  $b(i)$  of each pixel are obtained from the equation pair

$$\begin{cases} f_l(i) = k(i)g_l(i) + b(i) \\ f_d(i) = k(i)g_d(i) + b(i) \end{cases} \quad (8)$$

The systematic brightness error can be suppressed by

$$g(i) = \frac{f(i) - b(i)}{k(i)}. \quad (9)$$

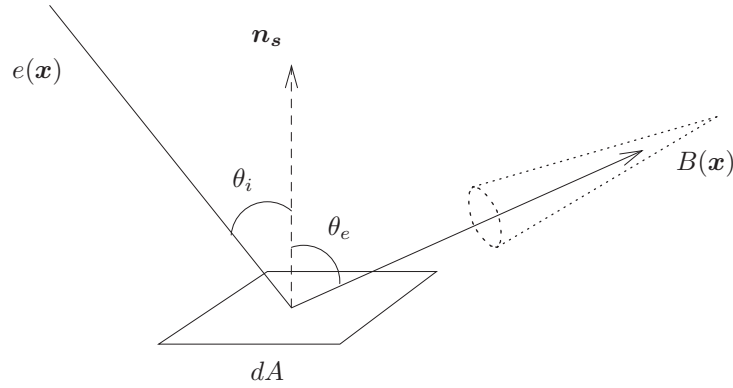


Figure 3: Radiometric image formation.

**3.**

The radiometric image formation is depicted in Fig. 3. The following definitions are made:

$e(\mathbf{x})$  Incident surface irradiance, which is the power per unit area of radiant energy falling on a surface. It is measured in units of watts per square meter ( $\text{W}/\text{m}^2$ ).

$B(\mathbf{x})$  Surface radiance, which is the power per unit foreshortened area emitted into a unit solid angle. It is measured in units of watts per square meter per steradian ( $\text{W}/\text{m}^2/\text{sr}$ ). Radiance can be a function of the viewing angle and the spectral wavelength and bandwidth.

$r(\mathbf{x})$  Surface reflectivity function (a.k.a. the reflectance, the reflection coefficient, or the bidirectional reflectance distribution function), which is the ratio of the radiant power per unit area per steradian reflected by the surface to the radiant power per unit area incident on the surface. It can be a function of the incident angle of the radiance, the viewing angle of the sensor, and the spectral wavelength and bandwidth.

$\theta_i$  Incidence angle.

$\theta_e$  Emittance angle.

The incident surface irradiance  $e(\mathbf{x})$  is captured by an infinitesimal surface element  $dA$ . The power is given by the projection  $e(\mathbf{x}) \cdot dA \cos \theta_i$ . The amount of surface illumination seen by a viewer is given by  $B(\mathbf{x}) \cdot dA \cos \theta_e$ . The surface reflectivity function relates these two quantities by the ratio

$$r(\mathbf{x}) = \frac{B(\mathbf{x}) \cos \theta_e}{e(\mathbf{x}) \cos \theta_i} \quad (10)$$

In this case we want to calculate the surface radiance as seen from a certain angle  $\theta_e$ , i.e. the foreshortened  $B(\mathbf{x}) \cdot \cos \theta_e$ . Lets call this simply  $B_e(\mathbf{x})$ . Then we get:

$$B_e(\mathbf{x}) = e(\mathbf{x}) \cos \theta_i r(\mathbf{x}). \quad (11)$$

Lambertian surfaces have the property that under uniform illumination they look equally bright from any direction. The amount of light reflected from a unit area goes down as the cosine of the viewing angle, but the amount of area seen in any solid angle goes up as the reciprocal of the cosine of the viewing angle. Because we relate the perceived brightness to radiant power per solid angle, Lambertian surfaces seem to have the same brightness independent of viewing angle. This leads to the surface reflectivity function being constant for Lambertian surfaces, i.e.  $r(\mathbf{x}) = r_0$ . This property is also sometimes given as the definition of a Lambertian surface.

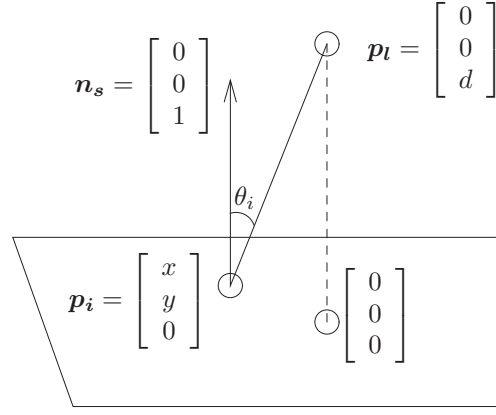


Figure 4: A Lambertian plane illuminated by a point source of light.

a) A Lambertian plane is illuminated by a point source of light at  $\mathbf{p}_l$ , Fig. 4. The reflectivity function  $r(\mathbf{p}_i)$  of the plane is constant,  $r(\mathbf{p}_i) = r_0$ . The irradiance  $e(\mathbf{p}_i)$  varies inversely as the square of the distance from the illuminated surface to the source (law of inverse squares),  $e(\mathbf{p}_i) = I_0/d_s^2$ .

$$\begin{aligned} B_e(\mathbf{p}_i) &= e(\mathbf{p}_i) \cos \theta_i r(\mathbf{p}_i) = \frac{I_0 \cos \theta_i r_0}{d_s^2} = \frac{I_0 r_0}{\|\mathbf{p}_l - \mathbf{p}_i\|^2} \frac{\mathbf{n}_s \cdot (\mathbf{p}_l - \mathbf{p}_i)}{\|\mathbf{n}_s\| \|\mathbf{p}_l - \mathbf{p}_i\|} \\ &= I_0 r_0 \frac{[0 \ 0 \ 1] [-x \ -y \ d]^T}{(x^2 + y^2 + d^2)^{3/2}} = \frac{I_0 r_0 d}{(x^2 + y^2 + d^2)^{3/2}} \end{aligned} \quad (12)$$

The brightest point on the plane is illuminated from normal direction ( $x = y = 0 \Rightarrow B_e(\mathbf{p}_i) = I_0 r_0/d^2$ ) and around this point the brightness decreases concentrically with decreasing illumination angle.

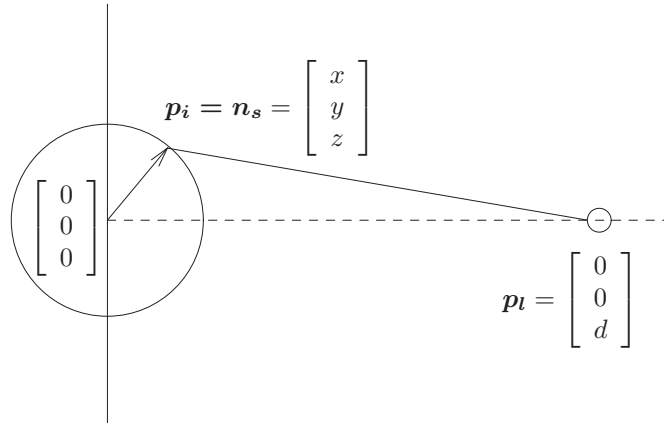


Figure 5: A Lambertian sphere illuminated by a point source of light.

b) A Lambertian sphere is illuminated by a point source of light  $\mathbf{p}_l$ , Fig. 5. Assume that the sphere has a unit radius, i.e.,  $\sqrt{x^2 + y^2 + z^2} = 1$ .

$$\begin{aligned} B_e(\mathbf{p}_i) &= I_0 r_0 \frac{[x \ y \ z] [-x \ -y \ d - z]^T}{(x^2 + y^2 + z^2 + d^2 - 2dz)^{3/2}} \\ &= I_0 r_0 \frac{-x^2 - y^2 - z^2 + dz}{(d^2 - 2dz + 1)^{3/2}} = \frac{I_0 r_0 (dz - 1)}{(d^2 - 2dz + 1)^{3/2}} \end{aligned} \quad (13)$$

The brightest point is illuminated from normal direction ( $z = 1 \Rightarrow B_e(\mathbf{p}_i) = I_0 r_0 / (d - 1)^2$ ). The brightness reaches zero at  $z = 1/d$  where the light hits the sphere tangentially.

4.

Two point sources of light illuminate a Lambertian plane, Fig. 6. The axis has been selected so that the lights are at points  $\mathbf{p}_l^A = [-e \ 0 \ d]^T$  and  $\mathbf{p}_l^B = [e \ 0 \ d]^T$ . The irradiance is  $e(\mathbf{p}_i) = I_0/d_s^2$ . The reflectivity function  $r(\mathbf{p}_i) = r_0$  is constant. The surface radiance at point  $\mathbf{p}_i = [x \ y \ 0]^T$  is then

$$\begin{aligned}
 B_e(\mathbf{p}_i) &= I_0 r_0 \left( \frac{\cos \theta_i^A}{d_A^2} + \frac{\cos \theta_i^B}{d_B^2} \right) = I_0 r_0 \left( \frac{\mathbf{n}_s \cdot (\mathbf{p}_l^A - \mathbf{p}_i)}{\|\mathbf{n}_s\| \|\mathbf{p}_l^A - \mathbf{p}_i\|^3} + \frac{\mathbf{n}_s \cdot (\mathbf{p}_l^B - \mathbf{p}_i)}{\|\mathbf{n}_s\| \|\mathbf{p}_l^B - \mathbf{p}_i\|^3} \right) \\
 &= I_0 r_0 \left( \frac{[0 \ 0 \ 1][(-e-x) \ (-y) \ d]^T}{((e+x)^2 + y^2 + d^2)^{3/2}} + \frac{[0 \ 0 \ 1][(e-x) \ (-y) \ d]^T}{((e-x)^2 + y^2 + d^2)^{3/2}} \right) \\
 &= I_0 r_0 \left( \frac{d}{((e+x)^2 + y^2 + d^2)^{3/2}} + \frac{d}{((e-x)^2 + y^2 + d^2)^{3/2}} \right) \tag{14}
 \end{aligned}$$

The radiance  $B_e(\mathbf{p}_i)$  is a sum of two radiances decreasing concentrically around the brightest point. The plane has two radiance maxima, at  $\mathbf{p}_i = [-e \ 0 \ 0]^T$  and  $\mathbf{p}_i = [e \ 0 \ 0]^T$ . The isoradiance curves on the plane are depicted in Fig. 7 ( $e = 1$ ).

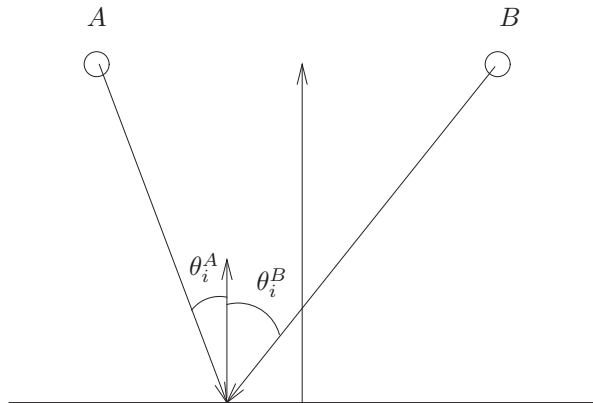


Figure 6: A Lambertian plane illuminated by two point sources of light.

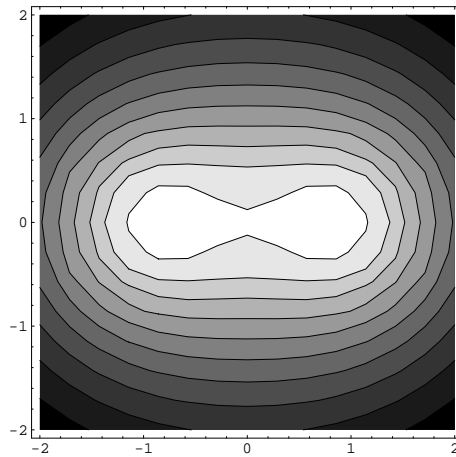


Figure 7: The variation of luminance at illumination by two point sources.

## T-61.5070 COMPUTER VISION, Exercise 2/08

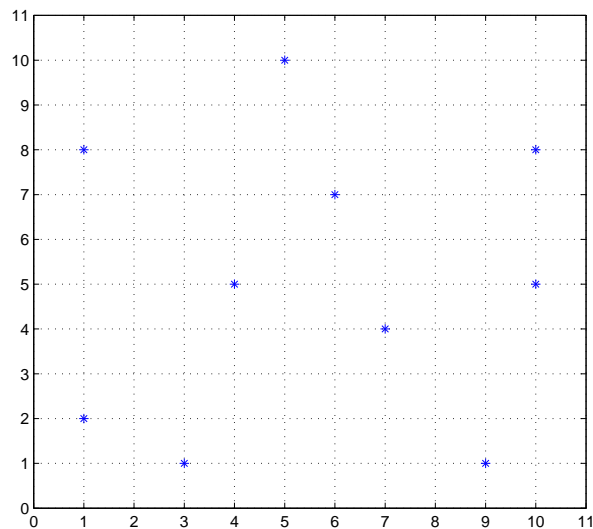
### Motivation

The purpose of this exercise is to be acquainted with the camera calibration, stereo vision, and surface modeling.

1. Determine the calibration parameters for the two cameras using these nine data points: 1, 3, 4, 5, 6, 8, 9, 10, 11. Set  $a_{34} = 1$ .

Point#	Camera A coordinates		Camera B coordinates		Object-centered coordinates (global, inches)		
	$x_a$	$y_a$	$x_i$	$y_i$	$x_0$	$y_0$	$z_0$
1	48	48	174	217	0	0	0
2	36	36	247	278			
3	36	48	223	283	1,72	1,95	0
4	48	36	217	223	0	0,05	0,24
5	36	24	280	275	0,64	3,04	0
6	24	24	339	356	2,76	5,88	0
7	48	24	257	232			
8	113	36	239	89	0	1,22	4,00
9	36	92	119	294	3,57	0,26	0
10	92	92	136	172	3,58	0	3,44
11	92	36	233	137	0	1,16	2,84

2. In a binocular animal vision system, assume a separation distance  $2h = 100\text{mm}$  and a focal length of an eye  $f = 50\text{mm}$ . Make a plot of the disparity as a function of distance. If the resolution of each eye is on order of 50 line pairs/mm, what is the useful range of the binocular system?
3. (a) Draw the Delaunay triangulation to the given datapoints.  
 (b) Construct the Voronoi diagram using the result from the previous item.  
 (c) Construct the convex hull of the given datapoints.



T-61.5070 COMPUTER VISION, Exercise 2/08

1.

Camera calibration is an essential stage in the utilization of a stereo imaging system. The relationship between pixel coordinates of 2-D image  $\mathbf{p}_i = [x_i \ y_i]^T$  and 3-D object  $\mathbf{p}_o = [x_o \ y_o \ z_o]^T$  must be determined. The relationship is nonlinear and therefore difficult to analyze.

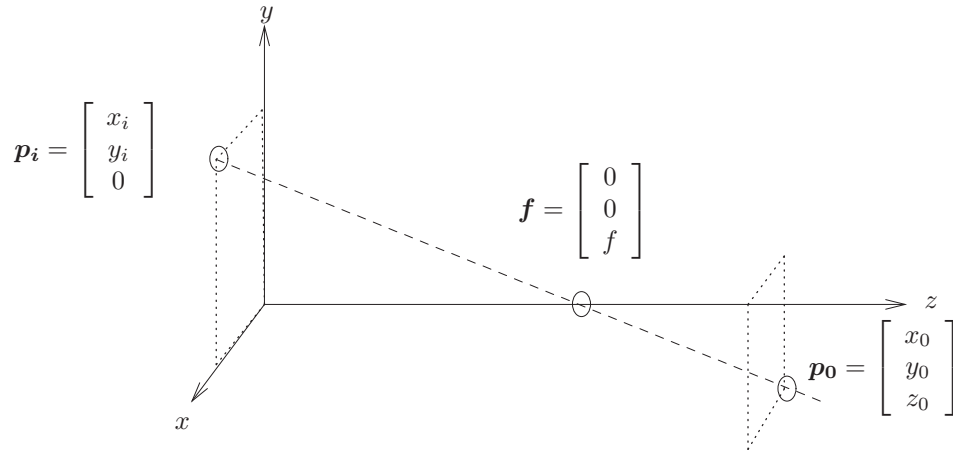


Figure 1: Pinpoint camera model.

For instance, in the simple **pinpoint camera model** the relationship is given by

$$\mathbf{f} - \mathbf{p}_i = k(\mathbf{p}_o - \mathbf{f}) = \tag{1}$$

$$\begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} = k \left( \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \right) \Leftrightarrow \begin{cases} x_i = -kx_o \\ y_i = -ky_o \\ k = \frac{f}{z_o - f} \end{cases} \tag{2}$$

$$\Leftrightarrow \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \frac{f}{f - z_o} x_o \\ \frac{f}{f - z_o} y_o \end{bmatrix} \tag{3}$$

It is often convenient to linearize Eq. 3. This can be accomplished easily by using **homogeneous coordinates**. A vector is transformed to homogeneous coordinates simply by expanding it by a unity element. Calculations are performed just as in conventional matrix algebra. When transforming back to the conventional coordinate system, the vector is scaled by the value of the additional coordinate. Thus, the division operation can be regarded as “implicitly implemented” with the additional coordinate, value of which generally changes during calculations but still serves as a “floating” scaling factor:

$$\begin{bmatrix} x \\ y \end{bmatrix} \hat{=} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} \tag{4}$$

The coordinates of a given image point are

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \hat{=} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \frac{f}{f - z_o} x_o \\ \frac{f}{f - z_o} y_o \\ 1 \end{bmatrix} \equiv \begin{bmatrix} fx_o \\ fy_o \\ f - z_o \end{bmatrix}. \tag{5}$$

The rightmost term in Eq. 5 can be expressed in matrix form as

$$\begin{bmatrix} fx_o \\ fy_o \\ f - z_o \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & -1 & f \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}. \quad (6)$$

Thus, the (pinpoint) camera model can be expressed as

$$\mathbf{p}_i = \mathbf{A}\mathbf{p}_o, \quad (7)$$

where the image coordinates are denoted by  $\mathbf{p}_i = [w_i x_i \ w_i y_i \ w_i]^T$ , object coordinates by  $\mathbf{p}_o = [w_o x_o \ w_o y_o \ w_o z_o \ w_o]^T$  and the imaging system (lenses etc.) by matrix  $\mathbf{A}$ .

The pinpoint camera model, that appears above as the simplicity of  $\mathbf{A}$ , is not useful for real sensor systems which require more complex transforms. During **camera calibration**, the elements in such transform matrix  $\mathbf{A}$  are estimated by studying sample object coordinates and resulting image points. If the matrix  $\mathbf{P}_o$  represents a set of object coordinates, the corresponding image points are obtained by  $\mathbf{P}_i = \mathbf{A}\mathbf{P}_o$ . In practice, the imaging system contains (slight) nonlinearities and the measurements of object and image points contain errors. In other words, there is no exact solution for the linear model represented by  $\mathbf{A}$ . An approximation of  $\mathbf{A}$  could be achieved by  $\mathbf{A} = \mathbf{P}_i \mathbf{P}_o^+$  (cf. definition of pseudoinverse below), but a more sophisticated method is presented in R. J. Schalkoff: *Digital Image Processing and Computer Vision*, pp. 29–30 (see also the course book Sec. 9.2.4/11.3.3 pp. 455/565–566):

Eq. 7 is expressed with homogeneous coordinates as

$$\begin{bmatrix} w_i x_i \\ w_i y_i \\ w_i \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}. \quad (8)$$

Letting  $a_{34} = 1$  in the homogeneous representation and expanding the matrix product yields

$$\begin{cases} w_i x_i = x_o a_{11} + y_o a_{12} + z_o a_{13} + a_{14} \\ w_i y_i = x_o a_{21} + y_o a_{22} + z_o a_{23} + a_{24} \\ w_i = x_o a_{31} + y_o a_{32} + z_o a_{33} + 1 \end{cases} \Leftrightarrow \begin{cases} x_i = x_o a_{11} + y_o a_{12} + z_o a_{13} + a_{14} - x_i x_o a_{31} - x_i y_o a_{32} - x_i z_o a_{33} \\ y_i = x_o a_{21} + y_o a_{22} + z_o a_{23} + a_{24} - y_i x_o a_{31} - y_i y_o a_{32} - y_i z_o a_{33} \end{cases} \quad (9)$$

For each of the  $N$  image-object coordinate pairs there are two equations, written in matrix product  $\mathbf{d} = \mathbf{Q}\mathbf{a}$ , where  $\mathbf{d} = [x_i^1 \ \cdots \ x_i^N \ | \ y_i^1 \ \cdots \ y_i^N]^T$ ,  $\mathbf{a} = [a_{11} \ a_{12} \ a_{13} \ a_{14} \ a_{21} \ a_{22} \ a_{23} \ a_{24} \ a_{31} \ a_{32} \ a_{33}]^T$ , and

$$\mathbf{Q} = \begin{bmatrix} x_o^1 & y_o^1 & z_o^1 & 1 & 0 & 0 & 0 & 0 & -x_o^1 x_o^1 & -x_o^1 y_o^1 & -x_o^1 z_o^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_o^N & y_o^N & z_o^N & 1 & 0 & 0 & 0 & 0 & -x_o^N x_o^N & -x_o^N y_o^N & -x_o^N z_o^N \\ - & - & - & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & x_o^1 & y_o^1 & z_o^1 & 1 & -y_o^1 x_o^1 & -y_o^1 y_o^1 & -y_o^1 z_o^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & x_o^N & y_o^N & z_o^N & 1 & -y_o^N x_o^N & -y_o^N y_o^N & -y_o^N z_o^N \end{bmatrix}$$

There is no solution for  $\mathbf{a} = \mathbf{Q}^{-1}\mathbf{d}$ , because  $\mathbf{Q}$  is not a square matrix. Therefore, vector  $\mathbf{a}'$  is searched for, which realizes the equation as well as possible,  $\mathbf{d} = \mathbf{Q}\mathbf{a}' + \mathbf{e}$ , where  $\mathbf{e}$  is an error term. The error can be minimized with the minimum squared error method. Find such  $\mathbf{a}'$  which minimizes the squared error

$$\mathbf{e}^T \mathbf{e} = (\mathbf{d} - \mathbf{Q}\mathbf{a}')^T (\mathbf{d} - \mathbf{Q}\mathbf{a}') = \mathbf{d}^T \mathbf{d} - \mathbf{a}'^T \mathbf{Q}^T \mathbf{d} - \mathbf{d}^T \mathbf{Q}\mathbf{a}' + \mathbf{a}'^T \mathbf{Q}^T \mathbf{Q}\mathbf{a}'. \quad (10)$$

Differentiation with respect to  $\mathbf{a}'$  and setting the result equal to zero gives

$$\begin{aligned}\frac{de^T e}{d\mathbf{a}'} &= -2\mathbf{Q}^T \mathbf{d} + 2\mathbf{Q}^T \mathbf{Q} \mathbf{a}' = 0 \\ &\Leftrightarrow \mathbf{Q}^T \mathbf{Q} \mathbf{a}' = \mathbf{Q}^T \mathbf{d} \\ &\Leftrightarrow \mathbf{a}' = (\mathbf{Q}^T \mathbf{Q})^{-1} \mathbf{Q}^T \mathbf{d} = \mathbf{Q}^+ \mathbf{d}\end{aligned}\quad (11)$$

Matrix  $\mathbf{Q}^+$  is the pseudoinverse of  $\mathbf{Q}$ . For the calibration of the two cameras the values calculated for  $\mathbf{Q}$  and  $\mathbf{d}$  are substituted into the pseudoinverse equation. This may be done with MATLAB.

```

Xo = [0 1.72 0 0.64 2.76 0 3.57 3.58 0]';
Yo = [0 1.95 0.05 3.04 5.88 1.22 0.26 0 1.16]';
Zo = [0 0 0.24 0 0 4.00 0 3.44 2.84]';
Xi = [48 36 48 36 24 113 36 92 92]';
Yi = [48 48 36 24 24 36 92 92 36]';
zer = [0 0 0 0 0 0 0 0 0]';
one = [1 1 1 1 1 1 1 1 1]';
q1 = [Xo Yo Zo one zer zer zer zer -Xi.*Xo -Xi.*Yo -Xi.*Zo];
q2 = [zer zer zer zer Xo Yo Zo one -Yi.*Xo -Yi.*Yo -Yi.*Zo];
Q = [q1 ; q2];
d = [Xi ; Yi];
'Calibration parameters for camera A'
a = inv(Q'*Q) * Q' * d

```

```

Xi = [174 223 217 280 339 239 119 136 233]';
Yi = [217 283 223 275 356 89 294 172 137]';
q1 = [Xo Yo Zo one zer zer zer zer -Xi.*Xo -Xi.*Yo -Xi.*Zo];
q2 = [zer zer zer zer Xo Yo Zo one -Yi.*Xo -Yi.*Yo -Yi.*Zo];
Q = [q1 ; q2];
d = [Xi ; Yi];
'Calibration parameters for the camera B'
a = inv(Q'*Q) * Q' * d

```

Calibration parameters for camera A:

-2.4548, -3.1710, 18.7095, 45.6199, 14.5518, -10.4855, 1.7594, 43.0276, 0.0076, -0.0260, 0.0189

Calibration parameters for camera B:

-20.0941, 27.1272, 7.6591, 192.1507, 21.2209, 4.9874, -32.7921 224.9605, 0.0073, -0.0247, 0.0231

## 2.

Binocular imaging geometry is represented in Sec. 9.2.5/11.5 pp. 457–460/573–583. A point in a 3-dimensional scene,  $P(x, y, z)$ , is projected onto the retinas of two eyes with  $2h$  separation, Fig. 2.

The projection point on the left retina is  $x_l$  and on the right  $x_r$ . In the depicted case,  $x_l$  is negative and  $x_r$  is positive. Disparity is the difference  $x_r - x_l$  and it is always non-negative.

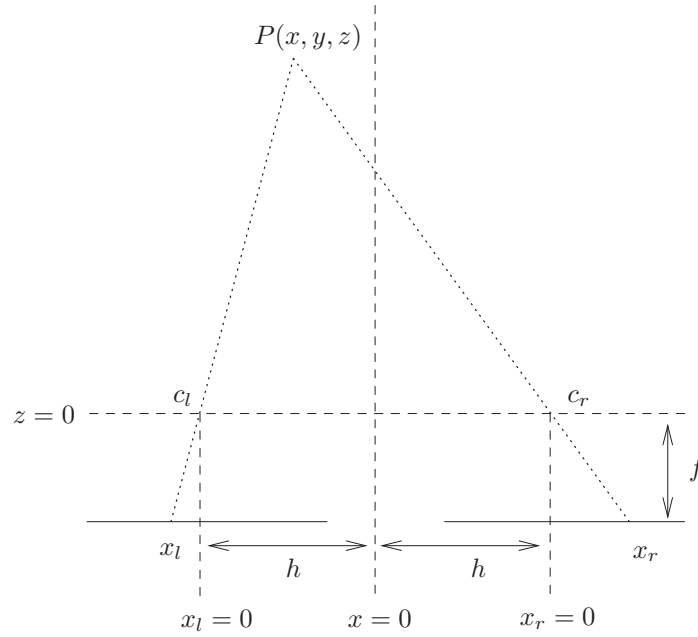


Figure 2: Stereo vision geometry (Sonka et al. 1993).

According to the geometry of similar triangles

$$\frac{-x_l}{f} = \frac{h - x_l + x}{z + f} \quad \text{and} \quad \frac{x_r}{f} = \frac{h + x_r - x}{z + f}. \quad (12)$$

Distance  $x$  is eliminated:

$$x = -\frac{hf + x_l z}{f} = \frac{hf - x_r z}{f}. \quad (13)$$

Disparity  $x_r - x_l$  is given by

$$z(x_r - x_l) = 2hf \Leftrightarrow x_r - x_l = \frac{2hf}{z}. \quad (14)$$

Resolution of retinal images is 50 lines/mm, separation of the eyes is 100, and focal length is 50 mm. Disparity is plotted as the function of object distance  $z$  in Fig. 3. The maximum object distance for a stereo image is

$$\frac{100 \text{ mm} \cdot 50 \text{ mm}}{(1/50) \text{ mm}} = 250 \text{ m}.$$

### 3.

a) In Delaunay triangulation the task is to find triangles that cover all data points in such a way that the circumcircle of any one triangle contains only the three points that are vertices of that particular triangle.

Let us start from the left edge on the given image. If we connect the first three points and draw a circle around them (Figure (a)), we notice that no other point lies inside the circle and thus our triangle is a valid one.

Then we add a new point and form a new triangle (Figure (b)). When we now draw a circle around our new triangle, the point marked with an arrow lies inside the circle, and thus our new triangle is not a valid one and we remove it.

We then select another point, form a new triangle, test it by drawing a circle around it and so on (Figure (c)).

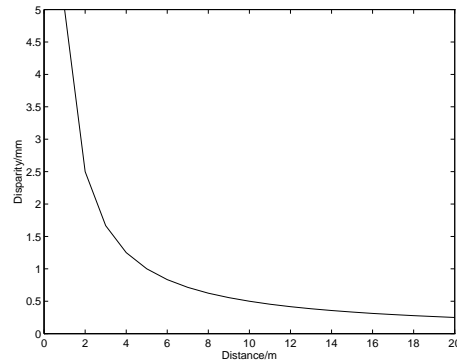
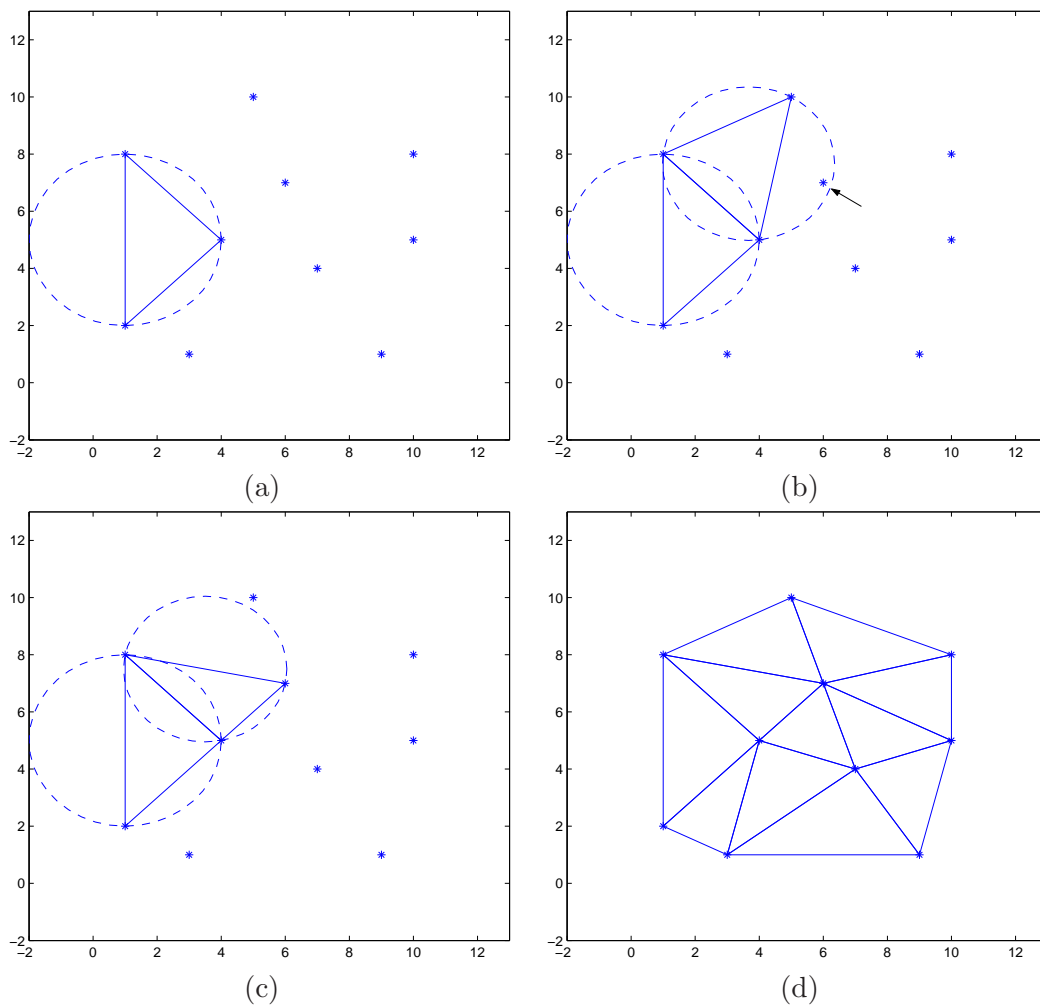


Figure 3: Disparity as a function of object distance.

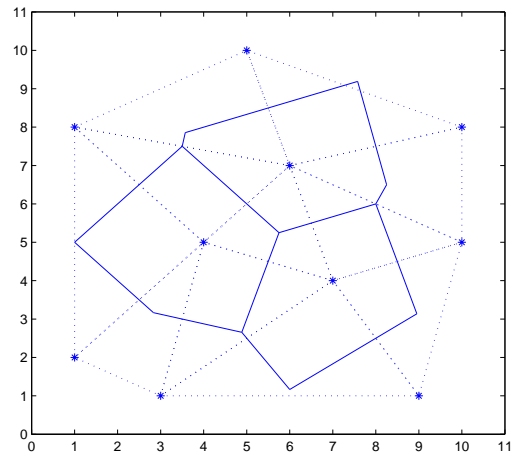
Finally we obtain the final Delaunay triangulation (Figure (d)). Notice that our solution is unique only if no more than three points lie on one circle!



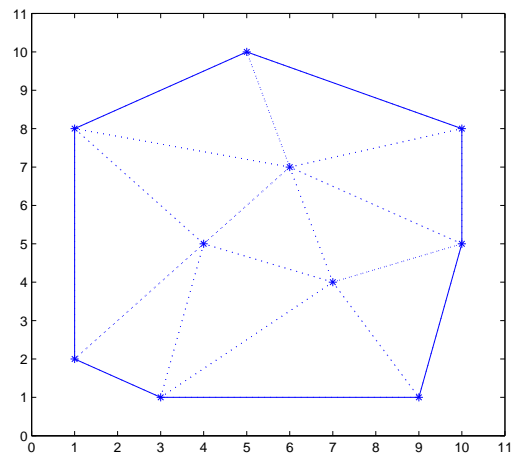
b) The Voronoi diagram on the given set of points is a set of convex polyhedra (or polygons in 2D space) that covers the whole space. Each given point  $M_i$  has one polyhedron  $V_i$ , and each polyhedron  $V_i$  covers all points in the space that are closer to the point  $M_i$  than to any other given point.

The Voronoi diagram can be constructed using the Delaunay triangulation. The midpoints of the sides of the triangles are recorded. The normal vectors are drawn to each triangle side through its midpoint (to both directions). The crossings of these normal vectors are calculated and convex polygons are formed. These polygons form the Voronoi diagram.

In the next figure the Voronoi diagram is drawn with solid lines and the Delaunay triangulation with dotted lines. Polygons that contain a point at infinity are not drawn.



c) The convex hull is obtained very easily, since it corresponds to the boundary of the set of points covered by triangles that were obtained by the Delaunay triangulation in a). In the next figure the convex hull is drawn with solid lines over the Delaunay triangulation (the dotted lines).



**T-61.5070 COMPUTER VISION, Exercise 3/08**

**Motivation**

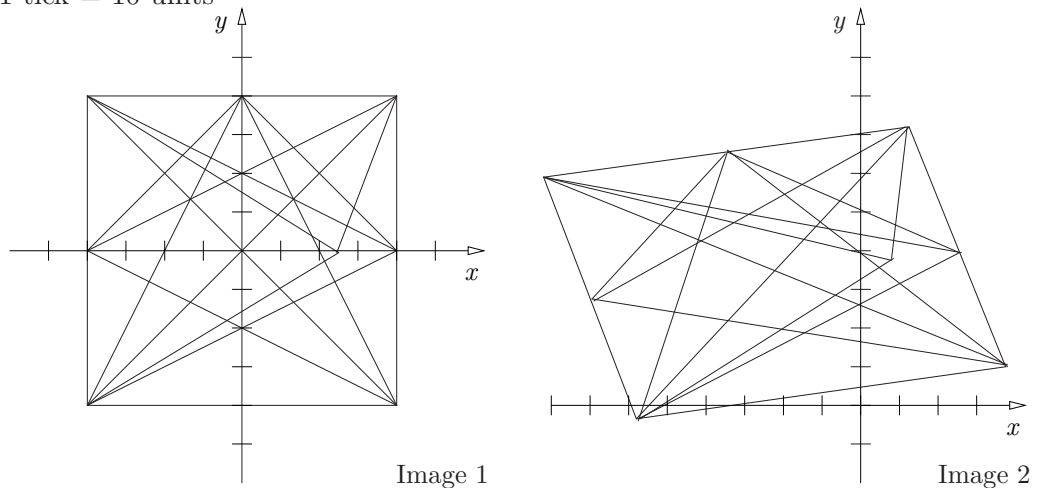
The purpose of this exercise is to be acquainted with data structures and the most usual geometric corrections.

1. Demonstrate how hierarchical search and matching work on the given example.

		IMAGE										
		0	0	0	0	0	0	0	0	0	0	0
TEMPLATE		0	0	0	0	0	0	9	0	0	0	0
		0	0	0	0	5	9	0	0	0	0	0
0	0	0	0	0	0	5	9	0	5	5	0	0
0	0	0	0	5	0	0	0	0	5	5	0	0
0	0	5	9	0	0	0	0	0	5	5	0	0
0	0	5	9	0	0	0	0	0	0	0	0	0
0	9	0	0	0	0	0	0	9	9	9	0	0
0	0	0	0	0	0	0	0	9	9	9	0	0
		0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	9	9	9	9	9	0
		0	0	0	0	0	0	0	0	0	0	0

2. Suggest an algorithm that can find an intersection between two regions represented as quadtrees. The intersection should be represented as a quadtree. Give an example!
3. Given the two binary images (below), choose appropriate control points and correct the coordinate system of the image 2 to the coordinate system of the image 1 using
  - (a) an affine transform,
  - (b) a 2D polynomial warp of 2nd order ( $N = 2$ ).

Scale: 1 tick = 10 units



4. (a) Using both analytical justification as well as graphical examples, show why the bilinear interpolation technique,

$$f(x', y') = (1-a)(1-b)f(x, y) + a(1-b)f(x+1, y) + (1-a)bf(x, y+1) + abf(x+1, y+1),$$

does not necessarily fit a plane to the interpolated intensity region.

- (b) What are the conditions on the local image intensities such that the bilinear interpolation technique results in fitting a plane to the image intensities?

## T-61.5070 COMPUTER VISION, Exercise 3/08

### 1.

Scene matching is represented in Sec. 5.4/6.4, pp. 190–194/237–241. Anil K. Jain represented hierarchical search in his book *Fundamentals of Digital Image Processing*, p. 407, as follows:

If the observed image is very large, we may first search a low-resolution-reduced copy using a likewise reduced copy of the template. If multiple matches occur, then the regions represented by these locations are searched using higher-resolution copies to further refine and reduce the search area. Thus the full-resolution region searched can be a small fraction of the total area. This method of *coarse-fine search* is also logarithmically efficient.

A coarse search is performed by reducing both the image and the template and by matching the reduced template to the image. The reductions are accomplished by replacing every  $2 \times 2$  neighborhood by its average. Values are rounded to the nearest integer. The reduced template is

$$h_r(i, j) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 7 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

The reduced image is

$$f_r(u, v) = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 7 & 1 & 1 & 0 \\ 0 & 2 & 0 & 3 & 3 & 0 \\ 0 & 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 2 & 0 & 0 \\ 0 & 0 & 2 & 5 & 5 & 0 \end{bmatrix}$$

Squared distance of the template from the image blocks yields a measure for their similarity

$$C(u, v) = \sum_{(i, j) \in V} [f(i + u, j + v) - h(i, j)]^2 = \quad (1)$$

$$\begin{bmatrix} 54 & 103 & 108 & 53 & 56 & 51 \\ 58 & 107 & \mathbf{11} & 113 & 52 & 52 \\ 62 & 94 & 136 & 84 & 32 & 64 \\ 58 & 84 & 49 & 100 & 68 & 63 \\ 54 & 102 & 71 & 126 & 88 & 59 \\ 54 & 73 & 80 & 67 & 38 & 79 \end{bmatrix}$$

The best-matching point, indicated in bold in the distance array, is suggestive for further evaluation. A finer search is done by matching the original template to this site in the original image.

$$\begin{array}{cccc} \vdots & \vdots & \vdots & \\ \cdots & 602 & 530 & 450 & \cdots \\ \cdots & 512 & \mathbf{91} & 662 & \cdots \\ \cdots & 422 & 555 & 752 & \cdots \\ \vdots & \vdots & \vdots & & \end{array}$$

The smallest squared distance for the template match, 91, was obtained when the left upper corner of the template was at position (2, 0) in the image.

2.

Quadtrees are represented in Sec. 3.3.2/4.3.2, pp. 51–52/108–109. Two sample images  $A$  and  $B$ , their intersection, and the corresponding quadtrees are depicted in Fig. 1.

The algorithm starts from the root node by processing its child nodes from left to right and by creating nodes in the intersection tree according to the following cases

1. Create a black leaf node if both counterpoint nodes are black leaf nodes.
2. Create a white leaf node if either or both counterpoint nodes are white leaf nodes.
3. Create a parent node if counterpoint nodes are a parent node, denoted by  $n$ , and a black leaf node. The child nodes of the parent node  $n$  are copied to the intersection tree.
4. If both counterpoint nodes are parent nodes then create a parent node and process iteratively the child nodes. If all child nodes result in white leaf nodes then replace the parent node with a white leaf node.

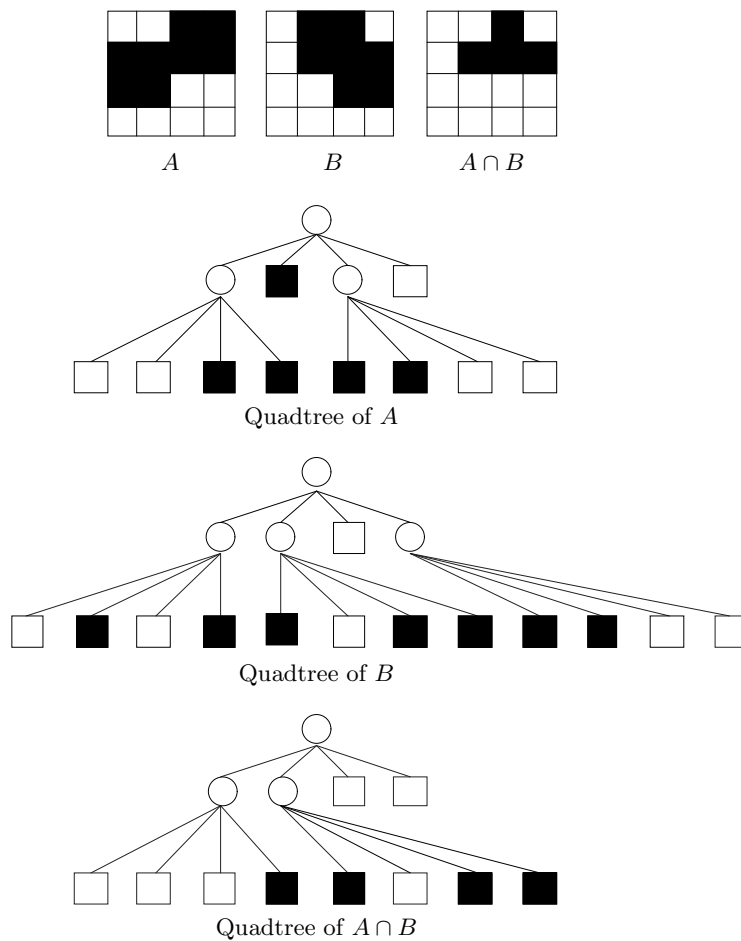


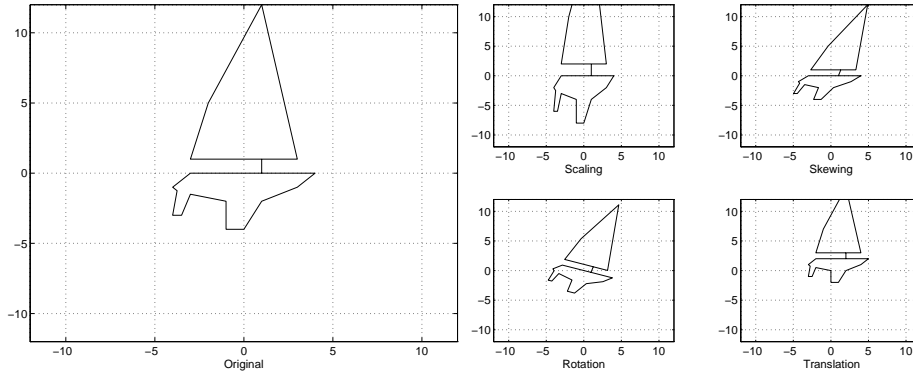
Figure 1: Two sample images, their intersection, and the quadtrees.

## 3.

a) Affine transformations (Sec. 4.2.1/5.2.1 p. 64/120) can be expressed in homogeneous coordinates by matrix product  $\mathbf{p}' = \mathbf{A}\mathbf{p}$ , where

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \text{ and } \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}. \quad (2)$$

The matrix  $\mathbf{A}$  is a combination of scaling  $\mathbf{A}_{sc}$ , skewing  $\mathbf{A}_{sw}$ , rotation  $\mathbf{A}_{rt}$ , and translation  $\mathbf{A}_{tr}$ ,  $\mathbf{A} = \mathbf{A}_{sc}\mathbf{A}_{sw}\mathbf{A}_{rt}\mathbf{A}_{tr}$ .



$$\mathbf{A}_{sc} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{A}_{sw} = \begin{bmatrix} 1 & \tan \phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{A}_{rt} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{A}_{tr} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

The order of different transformations is significant. Rotation followed by translation does not necessarily give the same result as translation followed by rotation.

Control points  $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N$  and their counterpart points  $\mathbf{p}'_1, \mathbf{p}'_2, \dots, \mathbf{p}'_N$  are selected to determine  $\mathbf{A}$ .

$$\mathbf{P}'_{3 \times N} = \mathbf{A}_{3 \times 3} \mathbf{P}_{3 \times N},$$

$$\mathbf{P} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \\ y_1 & y_2 & \dots & y_N \\ 1 & 1 & \dots & 1 \end{bmatrix}, \mathbf{P}' = \begin{bmatrix} x'_1 & x'_2 & \dots & x'_N \\ y'_1 & y'_2 & \dots & y'_N \\ 1 & 1 & \dots & 1 \end{bmatrix}. \quad (4)$$

Six elements in matrix  $\mathbf{A}$  are unknown. Therefore, three control points must be selected to solve for  $\mathbf{A} = \mathbf{P}'\mathbf{P}^{-1}$ . For instance, the following points may be chosen

```
P = [ -58 38 12; -4 10 72; 1 1 1];
Pc = [ -40 40 40; -40 -40 40; 1 1 1];
A = Pc * inv(P)
```

A =

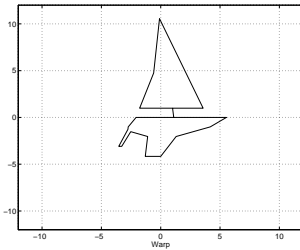
```
0.7853    0.3293    6.8651
-0.1773    1.2160   -45.4212
-0.0000    0.0000    1.0000
```

b) In the previous exercise, affine transforms were used for representation of geometric distortion functions. In this exercise, the functions are approximated by  $N$ th order 2-D polynomials, yielding the so-called *polynomial warp model*. Notice that no assumptions about the viewing geometry or about the character of geometric distortion are necessary. The degree of the 2-D polynomial is often selected empirically, or it is restricted by the number of available control points. The polynomial warp relationship is determined by

$$x' = \sum_{i=0}^N \sum_{j=0}^N \hat{k}_{ij}^1 x^i y^j \tag{5}$$

$$y' = \sum_{i=0}^N \sum_{j=0}^N \hat{k}_{ij}^2 x^i y^j. \tag{6}$$

An example of polynomial warp:



$$\begin{aligned} x' &= x + \frac{1}{10}x^2 - \frac{1}{1000}x^3 - \frac{1}{10}xy \\ y' &= y - \frac{1}{100}y^2 \end{aligned}$$

In our case,  $N$  is 2. The 18 coefficients  $\hat{k}_{ij}^{1,2}$  of the 2-D polynomial are estimated with two estimation equations from each control point. Therefore, at least nine control points are required. The transformation is given by  $\mathbf{p}' = \mathbf{K}\mathbf{w}$ , where

$$\begin{aligned} \mathbf{p}' &= [x' \ y']^T, \\ \mathbf{K} &= \begin{bmatrix} \hat{k}_{00}^1 & \hat{k}_{10}^1 & \hat{k}_{01}^1 & \cdots & \hat{k}_{22}^1 \\ \hat{k}_{00}^2 & \hat{k}_{10}^2 & \hat{k}_{01}^2 & \cdots & \hat{k}_{22}^2 \end{bmatrix}, \text{ and} \\ \mathbf{w} &= [1 \ x \ y \ xy \ x^2 \ y^2 \ x^2y \ xy^2 \ x^2y^2]^T. \end{aligned} \tag{7}$$

Matrix  $\mathbf{K}_{2 \times 9} = \mathbf{P}'_{2 \times 9} \mathbf{W}_{9 \times 9}^{-1}$ , where

$$\mathbf{P}' = \begin{bmatrix} x'_1 & \cdots & x'_N \\ y'_1 & \cdots & y'_N \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_N \\ \vdots & & \vdots \\ x_1^2 y_1^2 & \cdots & x_N^2 y_N^2 \end{bmatrix}.$$

The following points may be chosen, for instance

```
Px = [ -58 -70 -82 -24 -35 7 38 25 12]';
Py = [ -4 27 58 34 65 38 10 40 72]';
Pc = [ -40 -40 -40 0 0 24 40 40 40; ...
      -40 0 40 0 40 0 -40 0 40];
one = [1 1 1 1 1 1 1 1 1]';
W = [one Px Py Px.*Py Px.*Px Py.*Py Px.*Px.*Py ...
     Px.*Py.*Py Px.*Px.*Py.*Py]';
K = Pc * inv(W)
```

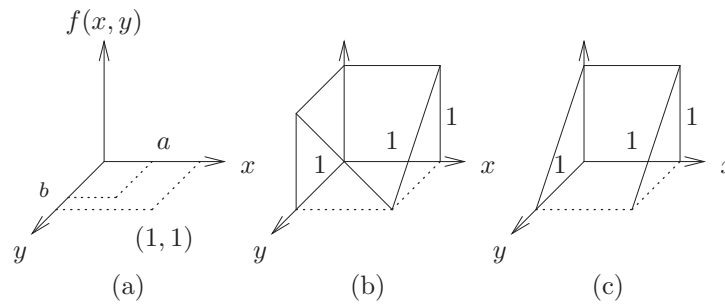


Figure 2: Bilinear interpolation

K =

Columns 1 through 6

57.5536	0.2388	-1.8436	0.0287	-0.0243	0.0212
-44.5636	-0.2115	1.1901	0.0033	-0.0008	0.0002

Columns 7 through 9

0.0012	-0.0003	-0.0000
0.0001	-0.0000	-0.0000

4.

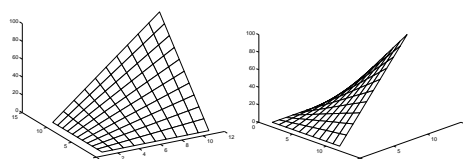
a) Bilinear interpolation is represented as linear interpolation in Sec. 4.2.2/5.2.2, p. 67/122–123. Assume that values of function  $f$  are known at points  $(0,0)$ ,  $(0,1)$ ,  $(1,0)$ , and  $(1,1)$ , Fig. 2. Bilinear interpolation yields the values of  $f(a,b)$ ,  $a \in [0,1]$  and  $b \in [0,1]$

$$\begin{aligned}
 f(a,b) = & (1-a)(1-b)f(0,0) \\
 & + a(1-b)f(1,0) \\
 & + (1-a)b f(0,1) \\
 & + abf(1,1)
 \end{aligned} \tag{8}$$

The Eq. 8 is a plane equation if it has the form  $f(a,b) = f_0 + f_1a + f_2b$ , i.e., the equation

$$\begin{aligned}
 f(a,b) = & f(0,0) + [f(1,0) - f(0,0)]a + [f(0,1) - f(0,0)]b \\
 & + [f(0,0) - f(1,0) - f(0,1) + f(1,1)]ab
 \end{aligned} \tag{9}$$

is a plane equation only if the factor of  $ab$  is equal to zero. For instance, if  $f(0,0) = f(1,0) = f(0,1) = 1$  and  $f(1,1) = 0$ , then  $f(a,b) = 1 - ab$ , which is not a plane, Fig. 2 (b). On the other hand, if  $f(0,0) = f(1,0) = 1$  and  $f(0,1) = f(1,1) = 0$ , then  $f(a,b) = 1 - b$ , which is a plane, Fig. 2 (c).



*Bilinearity: linear performance if either variable is fixed!*

b) The Eq. 9 is a plane equation if

$$f(0,0) + f(1,1) = f(1,0) + f(0,1), \quad (10)$$

i.e., when the control points  $f(0,0)$ ,  $f(1,1)$ ,  $f(1,0)$ , and  $f(0,1)$  are in the same plane.

**T-61.5070 COMPUTER VISION, Exercise 4/08****Motivation**

The purpose of this exercise is to brush up the most central edge detection and image enhancement methods.

1. Test the following edge detection algorithms for the given image:

- (a) Roberts operator
- (b) Sobel operator
- (c) Frein and Chen gradient masks

1	2	1	0	1	0	2
0	8	9	6	7	0	3
2	5	6	9	8	8	3
4	6	7	9	9	4	3
2	8	8	2	2	3	0
1	6	3	7	2	2	1
3	6	0	1	8	0	2
2	2	0	3	1	2	1

2. Show that median filtering based on an unsymmetrical neighbourhood may move the location of an edge.
3. Suggest an edge detection algorithm that can be used for volume images (3D) and for multispectral satellite images.
4. Assume a non-zero intensity distribution  $f(x)$ ,  $\{x \in [a, b]\}$ , in an image. Define a transform  $y = g(x)$  so that the distribution of  $y$  will be even in the range  $[0, c]$ . Derive an algorithm for histogram equalization based on that transform.

**T-61.5070 COMPUTER VISION, Exercise 4/08**

**1.**

Edge detecting spatial filters are described in Sec. 4.3.2/5.3.2, pp. 77–83/132–138. The convolution masks for *Roberts operator* are

$$h_1 : \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } h_2 : \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

*Sobel operator* defines two convolution masks suitable for the detection of horizontal and vertical edges:

$$h_1 : \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \text{ and } h_2 : \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

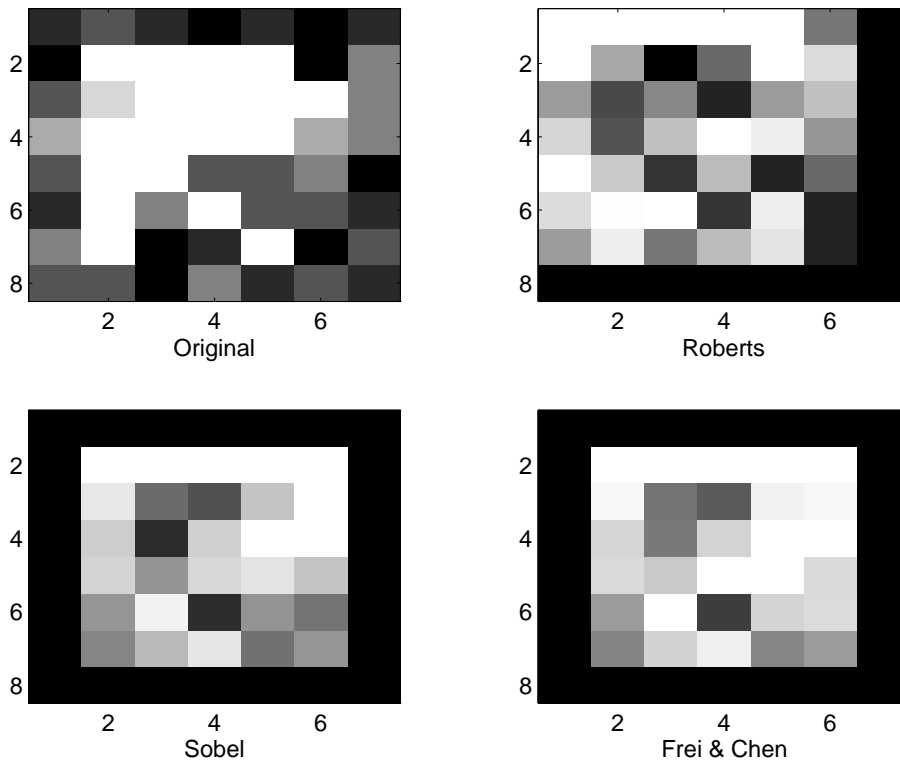
Frei and Chen used a set of nine orthogonal masks to detect edges and lines, or neighborhoods without edges and lines. Four of the nine masks are suitable for edge detection:

$$h_1 : \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}, h_2 : \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}, h_3 : \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}, \text{ and } h_4 : \begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}.$$

The output of mask  $h_k$  is given by

$$f_k(i, j) = \sum_{(m,n) \in R} h_k(i - m, j - n)g(m, n),$$

where  $g(m, n)$  is the image pixel and  $R$  defines the local neighborhood of the pixel  $f_k(i, j)$ . The  $L_2$  norm of the mask outputs,  $\sqrt{\sum_k f_k^2}$ , was used to describe edginess in the response images shown below. Sometimes, the simpler  $L_1$  norm ( $\sum_k |f_k|$ ) is used as an approximation although the systematic error increases with dimension.



## 2.

Median smoothing (pp. 74–76/129–131) ranks local pixel intensities and replaces the gray level of each pixel by the median of the gray levels in a neighborhood of that pixel. Define the support for a nonsymmetric median smoothing mask as



As shown by the following example, a nonsymmetric median filter may shift edge locations.

Original image		Filtered image
1 1 1 1 1		1 1 1 1 1
0 1 1 1 1		1 1 1 1 1
0 0 1 1 1	→	0 1 1 1 1
0 0 0 1 1		0 0 1 1 1
0 0 0 0 1		0 0 0 1 1

A symmetric median smoothing mask, e.g.,  $\begin{bmatrix} \square & \times & \square \end{bmatrix}$  preserves edge locations.

## 3.

Some methods for finding edges in multispectral images are described in Sec. 4.3.7/5.3.7, p. 94/147–148. The edge detector of Cervenka and Charvat, Eq. (4.68/5.63) in the textbook, is not appropriate for 3-D images because it does not take the gradient in  $z$ -direction into account. In 3-D data object boundaries are surfaces; edge elements in two dimensions become surface elements in three dimensions. The two-dimensional image gradient, when generalized to three dimensions, is the local surface normal. For a given volume image  $f(x, y, z)$ , the gradient of  $f$  is the vector defined by

$$\nabla f = \frac{\partial f}{\partial x} \mathbf{i} + \frac{\partial f}{\partial y} \mathbf{j} + \frac{\partial f}{\partial z} \mathbf{k},$$

where  $\mathbf{i} = [1 \ 0 \ 0]^T$ ,  $\mathbf{j} = [0 \ 1 \ 0]^T$ , and  $\mathbf{k} = [0 \ 0 \ 1]^T$  are the unit vectors. A simple measure of 3-D edginess is the magnitude of the surface gradient computed from differences approximating the first derivatives in the directions of  $x$ ,  $y$ , and  $z$ :

$$M = \sqrt{(\Delta_x f)^2 + (\Delta_y f)^2 + (\Delta_z f)^2},$$

where

$$\begin{aligned} \Delta_x f &= f(x+1, y, z) - f(x, y, z), \\ \Delta_y f &= f(x, y+1, z) - f(x, y, z), \\ \Delta_z f &= f(x, y, z+1) - f(x, y, z). \end{aligned}$$

This method is applied to multispectral images by forming the  $z$ -component using the  $n$  spectral bands  $f_0(x, y)$ ,  $f_1(x, y)$ ,  $\dots$ ,  $f_{n-1}(x, y)$ :

$$f(x, y, z) = f_z(x, y)$$

The brightness difference of the same pixel in two adjacent spectral bands,  $\Delta_z f$ , is very informative. (An example: The AVHRR instrument of NOAA weather satellites has five spectral channels. Some object characteristics are emphasized by differences between adjacent spectral bands, for example: The difference  $f_1(x, y) - f_0(x, y)$  serves the detection of vegetation and snow, and discrimination between land and sea. The difference  $f_3(x, y) - f_2(x, y)$  serves the detection of cirrus clouds and the discrimination between watercloud and surface, and between snow and cloud. The difference  $f_4(x, y) - f_3(x, y)$  serves the detection of precipitation clouds.)

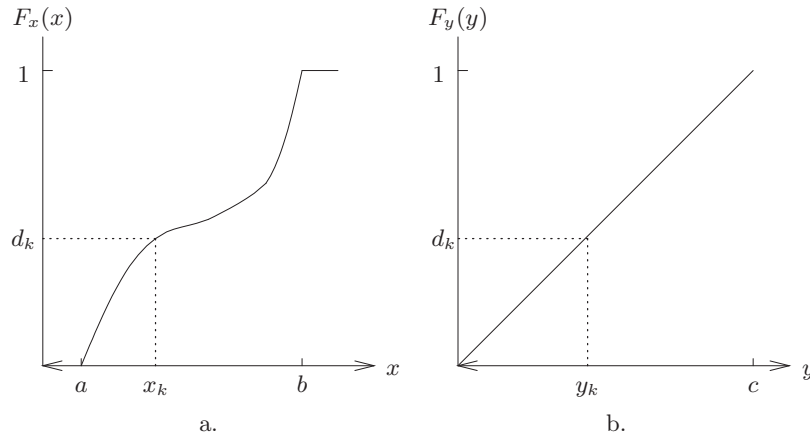


Figure 1: Cumulative distribution function of some  $f_x(x)$  (a.) and distribution function of constant density  $f_y(y)$  (b.).

4.

Histogram equalization is described in Sec. 4.1.2/5.1.2, pp. 59–62/115–118 in the textbook. A function  $y = g(x)$  is determined for transforming the non-uniform density function of  $x$ ,  $f_x(x)$ , into the constant density function of  $y$ ,  $f_y(y) = k$ . A cumulative distribution function of some  $f_x(x)$ ,  $F_x(x)$ , is depicted in Fig. 1.a. A distribution function of constant density  $f_y(y)$ ,  $F_y(y)$ , is depicted in Fig. 1.b. The intensity transform  $y = g(x)$  should be found which maps a value of  $x$  to a value of  $y$  so that  $F_y(y) = F_x(x)$ . Solve for  $y$ :

$$\begin{aligned}
 F_y(y) &= \int_0^y f_y(y)dy = \int_a^x f_x(x)dx = F_x(x). \\
 \int_0^y k dy &= \int_a^x f_x(x)dx \\
 ky &= \int_a^x f_x(x)dx \\
 y &= \frac{1}{k} \int_a^x f_x(x)dx.
 \end{aligned}$$

The normalization coefficient  $k$  can be determined from

$$F_y(c) = \int_0^c f_y(y)dy = 1 \Leftrightarrow kc = 1 \Leftrightarrow k = \frac{1}{c}.$$

Therefore,

$$y = g(x) = c \int_a^x f_x(x)dx = cF_x(x).$$

The histogram equalization algorithm is as follows:

1. Compute the histogram of intensity levels  $f_x(x)$  in the input image.
2. Sum  $f_x(x)$  to obtain the distribution function  $F_x(x)$ .
3. Use  $F_x(x)$  as the intensity transformation function  $g(x)$ ; that is,

$$y = cF_x(x),$$

where  $c$  is the largest intensity level in the transformed histogram.

## T-61.5070 COMPUTER VISION, Exercise 5/08

### Motivation

The purpose of this exercise is to be acquainted with morphological filtering.

1. Prove the following properties of dilation. See textbook, page 564/662.

(a)  $X \oplus B = B \oplus X$

(b)  $X \oplus (B \oplus D) = (X \oplus B) \oplus D$

(c)  $X \oplus B = \bigcup_{b \in B} X_b$

(d)  $X_h \oplus B = (X \oplus B)_h$

2. Skeletonize the objects in the given images by using the algorithm 6.8/8.8 (p. 267/365) and skeletonizing by maximal balls as in section 11.5.2/13.5.2.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3. Construct the quench functions for the four skeletons created in the previous exercise and try to reconstruct the original images using them. What is the appropriate type of ball in each case? Also create the ultimate erosions using the quench function.

**T-61.5070 COMPUTER VISION, Exercise 5/08****1.**

We will need the following definitions of dilation and translation.

The dilation of the point set  $X$  by the structuring element  $B$  is defined as

$$X \oplus B = \{p \in \mathcal{E}^2 \mid p = x + b \text{ for some } x \in X \text{ and } b \in B\}.$$

The translation of the point set  $X$  by the vector  $h$  is defined by

$$X_h = \{p \in \mathcal{E}^2 \mid p = x + h \text{ for some } x \in X\}.$$

a) Prove that dilation is commutative,  $X \oplus B = B \oplus X$ .

*Proof:*

$$\begin{aligned} X \oplus B &= \{p \mid p = x + b \text{ for some } x \in X, b \in B\} \\ &= \{p \mid p = b + x \text{ for some } x \in X, b \in B\} = B \oplus X \end{aligned}$$

b) Prove that dilation is associative,  $X \oplus (B \oplus D) = (X \oplus B) \oplus D$ .

*Proof:*

$p \in X \oplus (B \oplus D)$  if and only if there exists  $x \in X$ ,  $b \in B$ , and  $d \in D$  such that  $p = x + (b + d)$ .

$p \in (X \oplus B) \oplus D$  if and only if there exists  $x \in X$ ,  $b \in B$ , and  $d \in D$  such that  $p = (x + b) + d$ .

But  $x + (b + d) = (x + b) + d$  since addition is associative. Therefore,  $X \oplus (B \oplus D) = (X \oplus B) \oplus D$ .

c) Prove that dilation may also be expressed as a union of shifted point sets,  $X \oplus B = \bigcup_{b \in B} X_b$ .

*Proof:*

Suppose that  $p \in X \oplus B$ . Then for some  $x \in X$  and  $b \in B$ ,  $p = x + b$ . Hence,  $p \in (X)_b$  and therefore  $p \in \bigcup_{b \in B} X_b$ .

Suppose  $p \in \bigcup_{b \in B} X_b$ . Then for some  $b \in B$ ,  $p \in (X)_b$ . But  $p \in (X)_b$  implies there exists an  $x \in X$  such that  $p = x + b$ . Now by definition of dilation,  $x \in X$  and  $b \in B$ , and  $p = x + b$  imply  $p \in X \oplus B$ .

d) Prove that dilation is invariant to translation,  $X_h \oplus B = (X \oplus B)_h$ .

*Proof:*

$y \in X_h \oplus B$  if and only if for some  $z \in X_h$  and  $b \in B$ ,  $y = z + b$ . But  $z \in X_h$  if and only if  $z = x + h$  for some  $x \in X$ . Hence,  $y = (x + h) + b = (x + b) + h$ . Now by definition of dilation and translation  $y \in (X \oplus B)_h$ .



Assign  $R_{old} = R_{new}$  and continue.

$$S(R_{old}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_{old} - H_i(R_{old}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_o(S(R_{old})) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The intersection of  $H_o(S(R_{old}))$  and  $R_{old}$  is in bold. The union image is

$$R_{new} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The third iteration gives the same region. Therefore,  $R_{new}$  is a set of skeleton pixels of the region  $R$ .

### Skeletonizing by maximal balls

The skeleton can be created using maximal balls as explained in the textbook section 11.5.2/13.5.2. For image  $X$  with structural elements  $B$  (usually  $B_4$  or  $B_8$  balls), the skeleton  $S(X, B)$  is determined by the iterative set of operations

$$S(X, B) = \bigcup_{n=0}^N S_n(X, B),$$

where

$$\begin{aligned} S_n(X, B) &= (X \ominus nB) \setminus ((X \ominus nB) \circ B) \\ &= (X \ominus nB) \setminus ((X \ominus nB) \ominus B) \oplus B. \end{aligned}$$





$$((X \ominus B) \ominus B) \oplus B =$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$S_1(X, B) =$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$n = 2: S_2(X, B) = X \ominus 2B \setminus ((X \ominus 2B) \ominus B) \oplus B$$

$$X \ominus 2B =$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$((X \ominus 2B) \ominus B) \oplus B =$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$S_2(X, B) =$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$n = 3$ : The set  $X \ominus 3B$  is empty. Therefore,  $S_3(X, B)$  is empty and  $N = 2$ .

The skeleton of the second image is

$$S(X, B) = S_0(X, B) \cup S_1(X, B) \cup S_2(X, B) =$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

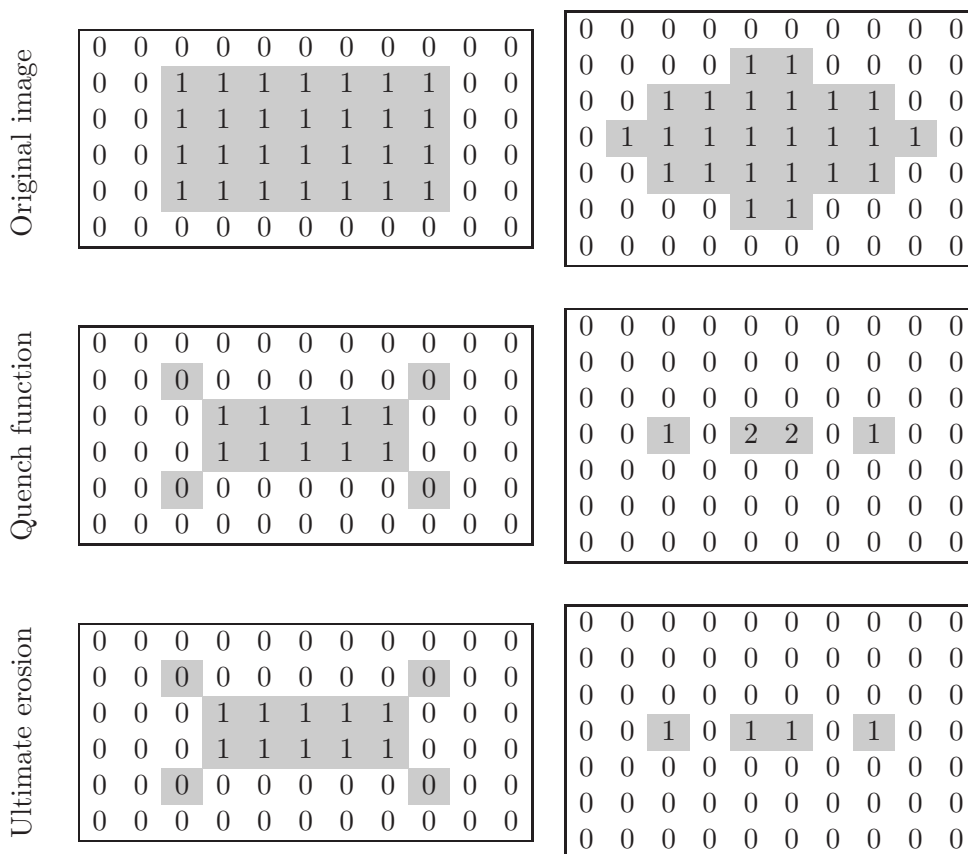
**Final skeletons produced**

Original image		
Maximal balls		

In comparison, the maximal balls algorithm produced a better skeleton for the first image, because it also included the “lines” for the corners. For the second image, the thinning algorithm was better, because the result from maximal balls was not contiguous, i.e. the transformation was not homotopic.

3.

The quench function and ultimate erosions are described in the book (section 11.5.4/13.5.4). Every point  $p$  of the skeleton by maximal balls has an associated ball which was a maximum ball at that point when performing the algorithm. The quench function  $q_X(p)$  is simply the radius of that ball at each point. We can define it to be zero outside of the skeleton. Using this function we can recreate the original object by placing a ball of the radius given by  $q_X(p)$  at each point. For the skeletons created by thinning there is no direct connection to the quench function because there are no maximal balls. The appropriate type of ball in this case is  $B_4$  since the skeletons were created with that.



The ultimate erosion of an image is the set of regional maxima of the quench function. The distinction between regional and local maxima are explained in the textbook (section 11.5.4/13.5.4). The regional maximum is a connected set of pixels  $M$  with the same value (e.g. grayscale value)  $h$ , such that every neighboring pixel of  $M$  has strictly lower value than  $h$ . Here we have used 4-neighborhoods.

**T-61.5070 COMPUTER VISION, Exercise 6/08****Motivation**

The purpose of this exercise is to be acquainted with some texture measures and some ways to characterize textures.

1. Generate a texture based on the given two level tree grammar:

$$G = [V_n, V_t, P, S], V_n = \{x, y, z\}, V_t = \{A_1, C_1\},$$

$$\begin{array}{c} x \rightarrow A_1 - y \text{ or } A_1 - y \\ | \\ x \end{array}$$

$P :$

$$y \rightarrow C_1 - z \text{ or } C_1$$

$$z \rightarrow A_1 - y \text{ or } A_1$$

2. Calculate the co-occurrence matrices for the given texture images using two different displacement vectors,  $\mathbf{d} = [1 \ 0]^T$  and  $\mathbf{d} = [0 \ 1]^T$ . Extract two features, contrast and entropy, from these co-occurrence matrices.

$$\begin{array}{cccccc} 2 & 0 & 0 & 0 & 2 & 0 \\ 1 & 2 & 0 & 1 & 0 & 2 \\ 0 & 2 & 1 & 0 & 2 & 0 \\ 0 & 0 & 2 & 1 & 0 & 2 \end{array}$$

3. Derive the  $3 \times 3$  Laws masks. What properties do different masks detect in images?
4. Suggest a texture measure based on the Fourier transform that is sensitive for texture directionality and coarseness.

**T-61.5070 COMPUTER VISION, Exercise 6/08**

**1.**

Grammars and languages are represented in Sec. 7.4.1/9.4.1, pp. 317–319/412–414. Methods for syntactic texture description are represented in Sec. 14.2/15.2, pp. 660–666/736–741. In the given grammar  $G = [V_n, V_t, P, S]$ ,  $V_n = \{x, y, z\}$  is the set of non-terminal symbols (variables),  $V_t = \{A_1, C_1\}$  is the set of terminal symbols, and  $P$  is the set of substitution rules:

$$\begin{array}{l}
 x \rightarrow A_1 - y \text{ or } A_1 - y \\
 \quad | \\
 \quad x \\
 P : \\
 y \rightarrow C_1 - z \text{ or } C_1 \\
 z \rightarrow A_1 - y \text{ or } A_1
 \end{array}$$

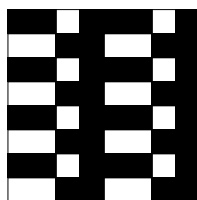
The starting symbol  $S$ , and the symbols  $A_1$  and  $C_1$  are not defined. Using  $x$  as the starting symbol produces, for example, the following chain:

$$\begin{array}{ccccc}
 & & A_1 - C_1 - z & & A_1 - C_1 - A_1 - y \\
 x \rightarrow A_1 - y & & | & \Rightarrow & | \\
 | & \Rightarrow & A_1 - y & \Rightarrow & A_1 - C_1 - z \\
 x & & | & & | \\
 & & x & & A_1 - y \\
 \\
 A_1 - C_1 - A_1 - C_1 & & A_1 - C_1 - A_1 - C_1 \\
 | & & | \\
 \Rightarrow A_1 - C_1 - A_1 - y & \Rightarrow & A_1 - C_1 - A_1 - C_1 \\
 | & & | \\
 A_1 - C_1 - z & & A_1 - C_1 - A_1 - C_1
 \end{array}$$

Therefore, if the symbols  $A_1$  and  $C_1$  are defined as

$$A_1 = \begin{array}{|c|} \hline \blacksquare \\ \hline \square \\ \hline \end{array} \text{ and } C_1 = \begin{array}{|c|} \hline \square \\ \hline \blacksquare \\ \hline \end{array}$$

then the grammar generates the following surface pattern:



**2.**

**Co-occurrence matrices**

The use of co-occurrence matrices in texture description is described in Sec. 14.1.2/15.1.2, pp. 651–653/723–725. The formalism of co-occurrence matrices represented below is somewhat different from that in the textbook. Assume an image function  $f(\mathbf{x})$  in which  $\mathbf{x} = [x \ y]^T$ ,  $x \in \{1, 2, \dots, X\}$  and  $y \in \{1, 2, \dots, Y\}$ .  $X$  and  $Y$  are the sizes of the image in  $x$ - and  $y$ -directions, respectively. The image is digitized using  $G$  gray levels  $g = 0, 1, \dots, G-1$ . Let  $\mathbf{d} = [\Delta x \ \Delta y]^T$  be a displacement vector, in which  $0 \leq \Delta x < X$  and  $-Y < \Delta y < Y$ . The co-occurrence matrix

$P_d(i, j)$  is defined as a matrix, the  $(i, j)$ th element of which is the number of appearances of gray level values  $i$  and  $j$  with separation and direction determined by the displacement  $\mathbf{d}$

$$P_d(i, j) = \#\{\mathbf{x} \mid f(\mathbf{x}) = i, f(\mathbf{x} + \mathbf{d}) = j\},$$

where  $\#$  represents the number of elements in the set. The matrix is normalized by dividing each entry with  $R = \sum_{i,j} P_d(i, j)$  which is the number of pixel pairs used in computing the matrix. The normalized matrix is  $p_d(i, j) = P_d(i, j)/R$ .

When the texture is coarse and the displacement  $\mathbf{d}$  is short in comparison with the texture elements, the examined pixel pairs  $\mathbf{x}$  and  $\mathbf{x} + \mathbf{d}$  have similar values. Therefore, the high values of  $p_d(i, j)$  are concentrated on the main diagonal. For a fine texture, the values of  $p_d(i, j)$  are spread out quite uniformly. If a texture is directional then the co-occurrence matrices obtained for displacements along the texture direction are characterized by high values along the main diagonal. Displacements in other directions produce co-occurrence matrices with scattered values.

**Exercise**

Define the two displacements as  $\mathbf{d}_{10} = [1 \ 0]^T$  and  $\mathbf{d}_{01} = [0 \ 1]^T$ . The first image and its two unnormalized co-occurrence matrices  $P_{10}(i, j)$  and  $P_{01}(i, j)$  are

$$\begin{array}{|c|c|c|} \hline 2 & 0 & 0 \\ \hline 1 & 2 & 0 \\ \hline 0 & 2 & 1 \\ \hline 0 & 0 & 2 \\ \hline \end{array} \quad \begin{array}{c|c|c|} P_{10} & 0 & 1 & 2 \\ \hline 0 & 2 & 0 & 2 \\ \hline 1 & 0 & 0 & 1 \\ \hline 2 & 2 & 1 & 0 \\ \hline \end{array} \quad \begin{array}{c|c|c|} P_{01} & 0 & 1 & 2 \\ \hline 0 & 2 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 \\ \hline 2 & 1 & 1 & 1 \\ \hline \end{array} .$$

The number of pixel pairs,  $R$ , used in computing the matrix is 8 for the displacement  $(1, 0)$  and 9 for the displacement  $(0, 1)$ . The corresponding contrast and entropy measures are

$$C = \sum_i \sum_j (i - j)^2 P_d(i, j) / R$$

$$C_{10} = [(0 - 2)^2 \cdot 2 + (1 - 2)^2 + (2 - 0)^2 \cdot 2 + (2 - 1)^2] / 8 = 18 / 8 = 2.25$$

$$C_{01} = [(0 - 1)^2 + (0 - 2)^2 + (1 - 0)^2 + (1 - 2)^2 + (2 - 0)^2 + (2 - 1)^2] / 9 = 12 / 9 \approx 1.33$$

$$E = - \sum_i \sum_j \frac{P_d(i, j)}{R} \log \frac{P_d(i, j)}{R}$$

$$E_{10} = - (3 \cdot \frac{2}{8} \log \frac{2}{8} + 2 \cdot \frac{1}{8} \log \frac{1}{8} + 4 \cdot 0 \log 0) = -\frac{6}{8} \log 2 + \log 8 \approx 0.68$$

$$E_{01} = - (1 \cdot \frac{2}{9} \log \frac{2}{9} + 7 \cdot \frac{1}{9} \log \frac{1}{9} + 1 \cdot 0 \log 0) = -\frac{2}{9} \log 2 + \log 9 \approx 0.89$$

Notice, that we have chosen to use  $\log_{10}$ , while the book uses  $\log_2$ . We have also defined  $0 \log 0 = 0$ , since  $\lim_{x \rightarrow 0} x \log x = 0$ .

The second image and its two unnormalized co-occurrence matrices are

$$\begin{array}{|c|c|c|} \hline 0 & 2 & 0 \\ \hline 1 & 0 & 2 \\ \hline 0 & 2 & 0 \\ \hline 1 & 0 & 2 \\ \hline \end{array} \quad \begin{array}{c|c|c|} P_{10} & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 4 \\ \hline 1 & 2 & 0 & 0 \\ \hline 2 & 2 & 0 & 0 \\ \hline \end{array} \quad \begin{array}{c|c|c|} P_{01} & 0 & 1 & 2 \\ \hline 0 & 0 & 2 & 3 \\ \hline 1 & 1 & 0 & 0 \\ \hline 2 & 3 & 0 & 0 \\ \hline \end{array} .$$

The corresponding contrast and entropy measures are

$$C_{10} = [(0-2)^2 \cdot 4 + (1-0)^2 \cdot 2 + (2-0)^2 \cdot 2] / 8 = 26/8 = 3.25$$

$$C_{01} = [(0-1)^2 \cdot 2 + (0-2)^2 \cdot 3 + (1-0)^2 \cdot 2 + (2-0)^2 \cdot 3] / 9 = 27/9 = 3.00$$

$$E_{10} = -\left(\frac{4}{8} \log \frac{4}{8} + 2 \cdot \frac{2}{8} \log \frac{2}{8}\right) = -\frac{12}{8} \log 2 + \log 8 \approx 0.45$$

$$E_{01} = -\left(2 \cdot \frac{3}{9} \log \frac{3}{9} + \frac{2}{9} \log \frac{2}{9} + \frac{1}{9} \log \frac{1}{9}\right) = -\frac{6}{9} \log 3 + \frac{2}{9} \log 2 + \log 9 \approx 0.57$$

### 3.

Laws masks are derived from three simple vectors:

$$L_3 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \quad (\text{center-weighted local average})$$

$$E_3 = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad (\text{edge detection})$$

$$S_3 = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \quad (\text{spot detection})$$

The following nine masks are obtained by multiplying these vectors with themselves or each other:

$$L_3^T \times L_3 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad L_3^T \times E_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad L_3^T \times S_3 = \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

$$E_3^T \times L_3 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad E_3^T \times E_3 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad E_3^T \times S_3 = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix}$$

$$S_3^T \times L_3 = \begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix} \quad S_3^T \times E_3 = \begin{bmatrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix} \quad S_3^T \times S_3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

These masks can be shown to span the space of  $3 \times 3$  neighborhoods, i.e., any  $3 \times 3$  array is a linear combination of them.

#### Properties of $3 \times 3$ Laws masks:

- $L_3 L_3$  ( $L_3^T \times L_3$ ) calculates the center-weighted average.
- $L_3 E_3$  and  $E_3 L_3$  are classical (center-weighted) edge detectors that detect vertical and horizontal edges. They are similar to Sobel masks.
- $L_3 S_3$  and  $S_3 L_3$  are (center-weighted) line detectors that will detect vertical and horizontal lines.  $E_3 E_3$  and  $S_3 S_3$  are also line detectors that will detect lines regardless of their orientation. They are similar to Frei-Chen line detector masks  $T_6$  and  $T_7$ .
- $E_3 S_3$  and  $S_3 E_3$  are included just for completeness, they don't detect any specific structures.

4.

The Fourier transform of region  $R$  in image  $f(x, y)$  is defined by

$$F(u, v) = \iint_R e^{-j2\pi(ux+vy)} f(x, y) dx dy$$

The real valued Fourier power spectrum, denoted  $|F(u, v)|^2$  is defined by

$$|F(u, v)|^2 = F(u, v)F^*(u, v).$$

The expression of Fourier power spectrum in polar coordinates  $|F(r, \theta)|^2$  where  $r = \sqrt{u^2 + v^2}$  and  $\theta = \arctan(v/u)$  provides measures for texture coarseness and directionality (see Sec. 14.1.1/15.1.1, p. 650/722).

The Fourier power spectrum of a coarse texture has largest values near the origin. The finer texture, the further the spectral peaks are from the origin. Therefore, the texture coarseness may be analyzed by examining rings  $\phi_r(r)$  with different radii  $r$ , or averages at regular intervals  $r_1 \leq r \leq r_2$

$$\phi_r(r) = \int_0^\pi |F(r, \theta)|^2 d\theta.$$

Directionality, lines or edges in direction  $\theta$ , results in high spectral values along the perpendicular direction  $\theta + \frac{\pi}{2}$ . Directionality may be analyzed by examining wedge-shaped regions  $\theta_1 \leq \theta \leq \theta_2$  in

$$\phi_\theta(\theta) = \int_0^\infty |F(r, \theta)|^2 dr.$$

**T-61.5070 COMPUTER VISION, Exercise 7/08****Motivation**

The purpose of this exercise is to be acquainted with image segmentation.

1. You have a histogram. Develop an algorithm to detect a threshold. Notice that the values in a histogram are often contaminated by noise.
  - (a) Test your algorithm on the given histogram.
  - (b) If it is possible to estimate the properties of the background and the objects, what kind of algorithm should you use?

Number of pixels	200	250	170	63	34	47	21	84	110	95	110
Gray-scale value	0	1	2	3	4	5	6	7	8	9	10

2. Create an edge-based segmentation for the given image. The object can be separated from the background using thresholding. Mention some methods for edge tracing.

1	2	0	1	0
1	5	2	5	1
0	6	7	5	1
1	2	7	5	1
1	7	2	7	0
0	1	1	0	1

3. How does “Papert’s turtle” manage following the edge in the the image in the previous exercise? How does the choice of pixel adjacency influence the results? Mention some other methods for edge following.

**T-61.5070 COMPUTER VISION, Exercise 7/08**

**1.**

Segmentation methods based on thresholds detected from image histograms are described in Sec. 5.1/6.1, pp. 127–131/179–183.

a) The given histogram is depicted in Fig. 1.a. The histogram has many local minima<sup>1</sup> and therefore it is smoothed by averaging over five-element neighborhoods, Fig. 1.b (see Sec. 2.3.2/2.3.2 in the book). To obtain the five-element average at the histogram edge, the edge bin is copied. The thresholds are now detected by finding all minima in the smoothed histogram.

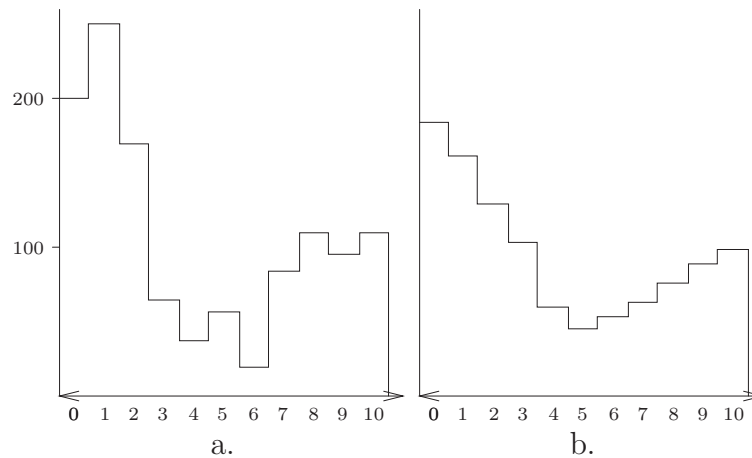


Figure 1: The given histogram (a.) and the smoothed histogram (b.)

The smoothed histogram has only one minimum at graylevel value 5.

b) If some regions in the image can be regarded as potential background or object pixels, then Algorithm 5.2/6.2 in the textbook (p. 129/181), iterative threshold selection, may be used. The step 1. is replaced with computation of  $\mu_B^0$  and  $\mu_O^0$  according to the background and object pixels. Set  $T^1 = (\mu_B^0 + \mu_O^0)/2$  and continue to step 2.

**2.**

Border tracing is described in Sec. 5.2.3/6.2.3, pp. 142–148/191–197. Border tracing methods are used to detect borders of regions obtained by thresholding. The image is segmented by thresholding to background and object regions:

$$\begin{cases} g(x, y) = 1, & \text{when } f(x, y) \geq 4 \\ g(x, y) = 0, & \text{when } f(x, y) < 4 \end{cases} \quad (1)$$

$f(x, y)$		$g(x, y)$
1 2 0 1 0	→	0 0 0 0 0
1 5 2 5 1		0 1 0 1 0
0 6 7 5 1		0 1 1 1 0
1 2 7 5 1		0 0 1 1 0
1 7 2 7 0		0 1 0 1 0
0 1 1 0 1		0 0 0 0 0

<sup>1</sup>'Noise' in the definition of the problem does not refer to image noise but to unevennesses in the histogram. Addition of noise to an image smoothes peaks in its histogram.

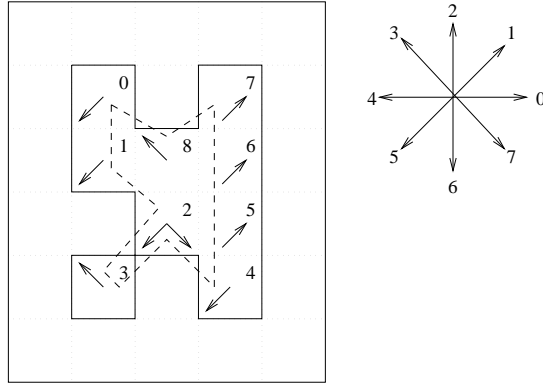


Figure 2: Finding the boundary in a binary image using inner boundary tracing. Direction notation in 8-connectivity is depicted on the right.

The textbook introduces four algorithms for border tracing:

Algorithm 5.8/6.7, inner boundary tracing (p. 142/192),

Algorithm 5.9/6.8, outer boundary tracing (p. 143/192) which is suitable for detection of shape, shape of perimeter or compact regions, for instance.

Algorithm 5.10/6.9, extended boundary tracing (p. 146/195–196) which defines a single common border between adjacent regions.

If object regions are not defined then Algorithm 5.11/6.10, border tracing in gray level images (p. 147/197), may be used. The border is represented by a simple path of high-gradient pixels.

Fig. 2 shows the path obtained by the inner boundary tracing algorithm in 8-connectivity. The arrows depict the direction at the beginning of each neighborhood search. The dashed lines depict the detected inner borders. The procedure is described in detail in the following.

1. Starting pixel  $P_0 = 0$  is found in the upper left corner of the region. Assign  $dir = 7$ .
2. Search the  $3 \times 3$  neighborhood in an anti-clockwise direction beginning in direction  $(7 + 6) \bmod 8 = 5$ .  $P_1 = 1$ ,  $dir = 6$ .
3. Search begins in direction  $(6 + 7) \bmod 8 = 5$ .  $P_2 = 2$ ,  $dir = 7$ .
4. Search begins in direction  $(7 + 6) \bmod 8 = 5$ .  $P_3 = 3$ ,  $dir = 5$ .
5. Search begins in direction  $(5 + 6) \bmod 8 = 3$ .  $P_4 = 2$ ,  $dir = 1$ .
6. Search begins in direction  $(1 + 6) \bmod 8 = 7$ .  $P_5 = 4$ ,  $dir = 7$ .
7. Search begins in direction  $(7 + 6) \bmod 8 = 5$ .  $P_6 = 5$ ,  $dir = 2$ .
8. Search begins in direction  $(2 + 7) \bmod 8 = 1$ .  $P_7 = 6$ ,  $dir = 2$ .
9. Search begins in direction  $(2 + 7) \bmod 8 = 1$ .  $P_8 = 7$ ,  $dir = 2$ .
10. Search begins in direction  $(2 + 7) \bmod 8 = 1$ .  $P_9 = 8$ ,  $dir = 5$ .
11. Search begins in direction  $(5 + 6) \bmod 8 = 3$ .  $P_{10} = 0$ ,  $dir = 3$ .
12. Search begins in direction  $(3 + 6) \bmod 8 = 1$ .  $P_{11} = 1$ ,  $dir = 6$ .

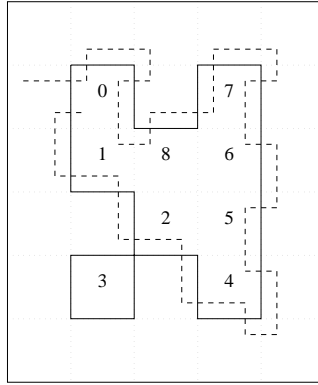


Figure 3: Finding the boundary in a binary image using Papert's turtle.

13. The current boundary element  $P_n = P_{11} = P_1$  and  $P_{n-1} = P_{10} = P_0$ . Stop.

The inner boundary is represented by pixels

$$P_0, \dots, P_{n-2} = P_0, \dots, P_9 = 0, 1, 2, 3, 2, 4, 5, 6, 7, 8. \quad (2)$$

### 3.

Edge following is described in sections 5.2.4–5/6.2.4–5, pp. 148–163/197–212. The textbook introduces three algorithms for edge following:

Algorithm 5.12/6.11: A-algorithm graph search (p. 149/198–199) which yields the optimal border connecting two image points, i.e., the starting and end points must be known. The optimal border minimizes some preselected cost function. The method is suitable for the detection of approximately straight contours.

Algorithm 5.13/6.12: Heuristic search for image borders (p. 157/206). Local edges are merged into chains on the basis of their magnitudes and directions. The use of the method does not require any knowledge about the border locations.

Algorithm 5.14/6.13: Boundary tracing as dynamic programming (p. 161/210) is suitable if the starting and end points of an edge are not known and, especially, if the cost functions are simple. The method is more efficient for some problems than the A-algorithm graph search (see the book, p. 161/210–211).

*Papert's turtle* searches for the boundaries of distinct image regions. The procedure is as follows:

1. Scan the image until a region pixel is encountered.
2. At a region pixel, turn left and step; else, turn right and step.
3. Terminate upon return to the starting pixel.

Regions with 4-connectivity are required for a consistent boundary detection; in an 8-connected region some parts may be missed. Some bookkeeping is necessary to generate an exact sequence of boundary pixels without duplications. Papert's turtle is a border tracing method rather than an edge following method; it requires no other knowledge than that about some object region. Fig. 3 shows the path traced out by the procedure. The detected border pixels with duplicates removed are

$$P_0, \dots, P_8 = 0, 1, 8, 7, 6, 5, 4, 2, 1. \quad (3)$$

The procedure missed one 8-connected border pixel in the lower left corner.

**T-61.5070 COMPUTER VISION, Exercise 8/08****Motivation**

This exercise covers image segmentation and starts to look at shape descriptions.

1. It is possible to use the Hough transform for edge detection and segmentation. It is, however, necessary to know something about the shape of the object. How can an edge that resembles a straight line be located by the Hough transform?
2. Propose a Hough transform for looking for circles in an image. How can you do the same with the generalized Hough transform?
3. Chain-numbers can be used to describe the shape of an object. Determine a chain-number for
  - (a) a circle,
  - (b) an octagon.

What is the Levenshtein distance between these figures?

4. Boundaries can be described by Fourier descriptors. Give an example and show that the Fourier description of boundaries makes it easier to recognize a given object.

**T-61.5070 COMPUTER VISION, Exercise 8/08**

**1.**

Hough transform is described in Sec. 5.2.6/6.2.6, pp. 163–173/212–221. First, the edges in the image are detected, e.g., using gradient operators. Algorithm 5.15/6.14, curve detection with the Hough transform (p. 167/217), may then be applied to the edge image. The arbitrary curve equation,  $f(\mathbf{x}, \mathbf{a}) = 0$ , is now a line equation

$$y = kx + q. \tag{1}$$

1. Quantize the parameter space  $(k', q')$  within the limits of parameters  $k$  and  $q$  using, for instance, the definitions

$$k' = \min k, \min k + \Delta k', \dots, \max k - \Delta k', \max k;$$

$$q' = \min q, \min q + \Delta q', \dots, \max q - \Delta q', \max q.$$

2. Form a two-dimensional accumulator array  $A(k', q')$  whose size is the number of elements in  $k'$  multiplied by the number of elements in  $q'$ :  $\# \{A(k', q')\} = \# \{k'\} \times \# \{q'\}$ . Set all elements of  $A(k', q')$  to zero.

3. For each point  $(x, y)$  in the edge image, form the parameter line equation

$$q = -xk + y, \tag{2}$$

and increase accumulator cells  $A(k', q')$  for all  $k'$  and  $q'$  that are hit by the line of Eq. 2 (see Fig. 1):

$$A(k', q') = A(k', q') + 1. \tag{3}$$

4. Local maxima in the array  $A(k', q')$  correspond to lines that are present in the original image.

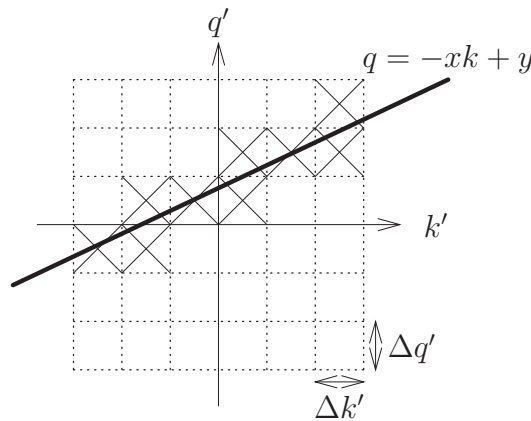


Figure 1: Quantized parameter space.

**2.**

The Hough transform for the detection of circles is explained in the textbook on p. 168/217. The general curve equation,  $f(\mathbf{x}, \mathbf{a}) = 0$ , becomes the circle equation

$$(x - x_0)^2 + (y - y_0)^2 = r^2. \tag{4}$$

1. Form a quantized parameter space  $(x'_0, y'_0, r')$ .
2. Form a three-dimensional accumulator array  $A(x'_0, y'_0, r')$ . Set all elements of  $A(x'_0, y'_0, r')$  to zero.
3. For each point  $(x, y)$  in the edge image, update those accumulator cells  $A(x'_0, y'_0, r')$  for which the point  $(x'_0, y'_0)$  is at the distance  $r'$  from the point  $(x, y)$ .
4. The local maxima in the array  $A(x'_0, y'_0, r')$  correspond to the circles in the original image.

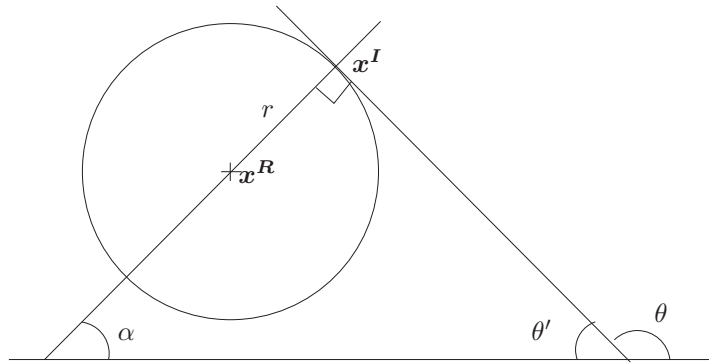


Figure 2: Geometry for the description of a circle with an R-table

The generalized Hough transform is explained on pp. 171–172/219–220. Follow the Algorithm 5.16/6.15:

1. An R-table for a sample circle with radius  $r$  should be constructed. A natural reference point for a circle is the center

$$\mathbf{x}^R = [x_0 \ y_0]^T. \quad (5)$$

A line that goes through the center in direction  $\alpha$  is a normal of the circle, Fig. 2. The tangent at the intersection point,  $\mathbf{x}^I$ , gives the edge direction  $\theta$ . The directions  $\alpha$  and  $\theta$  are related by the equation

$$\alpha = 90^\circ - \theta' = 90^\circ - (180^\circ - \theta) = \theta - 90^\circ. \quad (6)$$

Because  $\alpha$  is a simple function of  $\theta$  and  $r$  is constant for all  $\theta$  they need not be tabulated. The R-table would thus contain only one element: the radius of the sample circle. Therefore, the description of a circle does not require an R-table.

2. To detect circles with different radii, the radius  $r$  is replaced by a set of quantized radii  $r_i \in ]0, R]$ . Form an array  $A(x_0, y_0, r_i)$  which represents the potential reference points. Set the values in the array  $A$  to zero.
3. Determine the edge direction  $\theta$  for each edge pixel  $(x, y)$  in the thresholded gradient image. The potential reference points  $\mathbf{x}^R$  are on the line in the direction  $\theta - 90^\circ$  at the distances  $r_i$ ; there are two potential reference points for each  $r_i$ . Increase all  $A(x_0, y_0, r_i)$ , where

$$x_0 = x + r_i \cos(\theta - 90^\circ), \quad y_0 = y + r_i \sin(\theta - 90^\circ) \quad (7)$$

or

$$x_0 = x - r_i \cos(\theta - 90^\circ), \quad y_0 = y - r_i \sin(\theta - 90^\circ). \quad (8)$$

4. The centers and radii of the circles are given by the local maxima in the array  $A$ .

Compare this result with Eq. (5.28/6.27) in the textbook. The procedure was reduced into the Hough transform which takes the edge direction into account. The Hough transform is a special case of the generalized Hough transform.

3.

The chain-number refers to the normalized chain code which is explained in Sec. 6.2.1/8.2.1, pp. 236/335–336 in the textbook. Fig. 3 represents a spatially quantized circle and octagon. The chain-number of the circle is

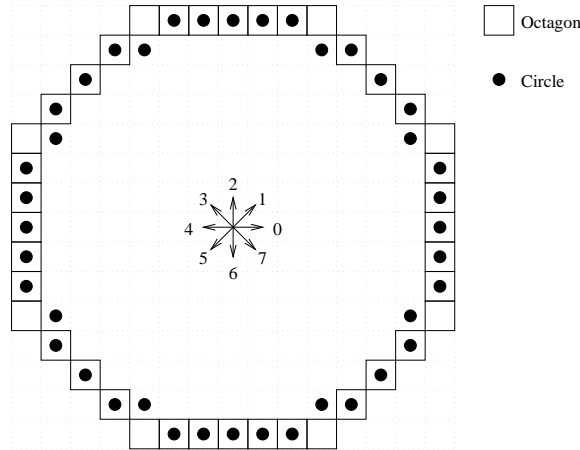


Figure 3: Circle and octagon for chain code formation.

1000070776766665655454444343323222212110.

The chain-number of the octagon is

1000000777766666655554444443333222222111.

Levenshtein distance is defined as the smallest number of deletions, insertions, and substitutions necessary to convert one string into the other. Eight substitutions are needed to convert the chain-number of the circle to that of the octagon.

4.

Fourier transforms of boundaries are explained in Sec. 6.2.3/8.2.3, pp. 240–242/339–341. Two rotated and translated versions of the same irregular shape are depicted in Fig. 4.

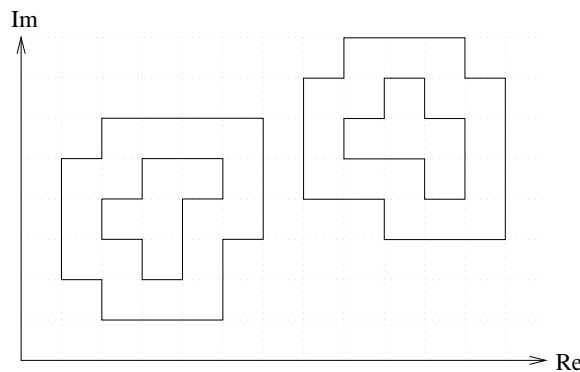


Figure 4: A shape with two different rotations and translations.

The boundary of the image on the left is given by the coordinates

$$B_A = (3, 6), (4, 6), (5, 6), (6, 6), (6, 5), (6, 4), (5, 4), (5, 3), \dots$$

$$(5, 2), (4, 2), (3, 2), (3, 3), (2, 3), (2, 4), (2, 5), (3, 5). \quad (9)$$

Similarly, the boundary of the object on the right is given by the coordinates

$$B_B = (9, 8), (10, 8), (11, 8), (11, 7), (12, 7), (12, 6), (12, 5), (12, 4), \dots \\ (11, 4), (10, 4), (10, 5), (9, 5), (8, 5), (8, 6), (8, 7), (9, 7). \quad (10)$$

The coordinates define for both images a set of sixteen complex numbers. The Fourier descriptors of the boundaries are computed from the discrete Fourier transform of the complex numbers. Absolute values of the Fourier descriptors are computed in *Mathematica*:

```
In[21]:= zA = {3 + 6 I, 4 + 6 I, 5 + 6 I, 6 + 6 I,
6 + 5 I, 6 + 4 I, 5 + 4 I, 5 + 3 I,
5 + 2 I, 4 + 2 I, 3 + 2 I, 3 + 3 I,
2 + 3 I, 2 + 4 I, 2 + 5 I, 3 + 5 I}
```

```
In[22]:= zB = {9 + 8 I, 10 + 8 I, 11 + 8 I, 11 + 7 I,
12 + 7 I, 12 + 6 I, 12 + 5 I, 12 + 4 I,
11 + 4 I, 10 + 4 I, 10 + 5 I, 9 + 5 I,
8 + 5 I, 8 + 6 I, 8 + 7 I, 9 + 7 I}
```

```
In[23]:= Abs[Fourier[zA]] // N
```

```
Out[23]= {22.9837, 7.90317, 0.270598, 0.587938,
> 0.707107, 0.527311, 0.270598, 0.13795,
> 0.5, 0.390655, 0.653281, 0.392847,
> 0., 0.330522, 0.653281, 0.69352}
```

```
In[24]:= Abs[Fourier[zB]] // N
```

```
Out[24]= {47.0771, 7.90317, 0.270598, 0.587938,
> 0.707107, 0.527311, 0.270598, 0.13795,
> 0.5, 0.390655, 0.653281, 0.392847,
> 0., 0.330522, 0.653281, 0.69352}
```

Excluding the DC-components, the Fourier descriptors for the two image boundaries are identical. The example demonstrates that Fourier descriptors are suitable for rotation and translation invariant object recognition.

**T-61.5070 COMPUTER VISION, Exercise 9/08****Motivation**

The purpose of this exercise is to be acquainted with shape description.

1. Give an example of a case where Euler's number is a proper feature.
2. Determine for the given images
  - (a) the moments  $m_{00}$ ,  $m_{01}$ ,  $m_{10}$ ,  $m_{11}$ ,  $m_{20}$ ,  $m_{02}$
  - (b) the center of gravity  $x_0$  and  $y_0$
  - (c) the central moments  $\mu_{00}$ ,  $\mu_{01}$ ,  $\mu_{10}$ ,  $\mu_{11}$ ,  $\mu_{20}$ ,  $\mu_{02}$

1	1	1	1	1	0	0	0	1	0
0	0	0	1	0	0	0	1	1	0
0	0	1	0	0	0	1	0	1	0
0	0	0	1	0	1	0	0	1	0
0	0	0	0	1	1	1	1	1	1
1	0	0	0	1	0	0	0	1	0
0	1	1	1	0	0	0	0	1	0

3. Calculate the PCA-transform to the following sample points:

$$\mathbf{x}_1 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \quad \mathbf{x}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{x}_5 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \mathbf{x}_6 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

- (a) Transform the sample points to one dimension. What is the mean square error?
- (b) Calculate the elongatedness of the region that is spanned by the sample points.

## T-61.5070 COMPUTER VISION, Exercise 9/08

### 1.

The Euler number (Sec. 6.3.1/8.3.1, p. 256/354),  $E$ , defines a topological property of an image: it is the difference between the number of connected components,  $C$ , and the number of holes,  $H$ .

$$E = C - H. \quad (1)$$

The Euler number is a rather general descriptor but as an additional feature it is often useful in characterizing regions in a scene. The Euler number is not affected by rotation, stretching, or scaling. This property is useful in character recognition, for instance. Fig. 1 shows four character images with different Euler numbers.

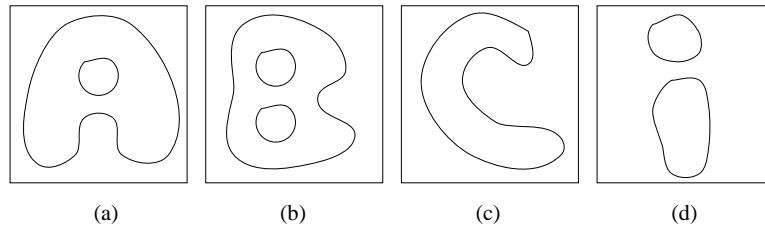


Figure 1: Topological description with the Euler number. (a)  $E = 1 - 1 = 0$ . (b)  $E = 1 - 2 = -1$ . (c)  $E = 1 - 0 = 1$ . (d)  $E = 2 - 0 = 2$ .

### 2.

Moments are defined by

$$m_{ij} = \sum_x \sum_y x^i y^j f(x, y). \quad (2)$$

The center of gravity is expressed by moments:

$$x_0 = \frac{\sum \sum x f(x, y)}{\sum \sum f(x, y)} = \frac{m_{10}}{m_{00}} \quad (3)$$

$$y_0 = \frac{\sum \sum y f(x, y)}{\sum \sum f(x, y)} = \frac{m_{01}}{m_{00}} \quad (4)$$

Central moments are defined by

$$\mu_{ij} = \sum_x \sum_y (x - x_0)^i (y - y_0)^j f(x, y). \quad (5)$$

They can be expressed as moments as shown in the following examples:

$$\begin{aligned} \mu_{00} &= \sum \sum f(x, y) = m_{00} \\ \mu_{01} &= \sum \sum y f(x, y) - \sum \sum y_0 f(x, y) = m_{01} - (m_{01}/m_{00})m_{00} = 0 \\ \mu_{10} &= \sum \sum x f(x, y) - \sum \sum x_0 f(x, y) = m_{10} - (m_{10}/m_{00})m_{00} = 0 \\ \mu_{11} &= m_{11} - m_{01}m_{10}/m_{00} \\ \mu_{20} &= m_{20} - m_{10}^2/m_{00} \\ \mu_{02} &= m_{02} - m_{01}^2/m_{00}. \end{aligned} \quad (6)$$

		Image 1	Image 2
a)	$i$	$j$	$m_{ij}$
	0	0	$\sum \sum f(x, y) = 14$
	0	1	$\sum \sum yf(x, y) = 38$
	1	0	$\sum \sum xf(x, y) = 32$
	1	1	$\sum \sum xyf(x, y) = 88$
	2	0	$\sum \sum x^2f(x, y) = 98$
	0	2	$\sum \sum y^2f(x, y) = 188$

	Image 1	Image 2
b)	$x_0 = 2\frac{2}{7}$	$x_0 = 2\frac{3}{14}$
	$y_0 = 2\frac{5}{7}$	$y_0 = 3\frac{1}{14}$

		Image 1	Image 2
c)	$i$	$j$	$\mu_{ij}$
	0	0	14
	0	1	0
	1	0	0
	1	1	8/7
	2	0	174/7
	0	2	594/7

**3.**

PCA (principal component analysis) -transform is given by

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m}_x) \quad (\text{the rows of } \mathbf{A} \text{ are the eigenvectors of } \mathbf{C}_x)$$

Mean:

$$\mathbf{m}_x = \frac{1}{M} \sum_{k=1}^M \mathbf{x}_k$$

Covariance matrix:

$$\mathbf{C}_x = \frac{1}{M} \sum_{k=1}^M \mathbf{x}_k \mathbf{x}_k^T - \mathbf{m}_x \mathbf{m}_x^T$$

The mean of the six sample points is now

$$\mathbf{m}_x = \frac{1}{6} \begin{pmatrix} -2 - 1 + 0 + 0 + 1 + 2 \\ 0 + 2 + 3 + 1 + 2 + 4 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

and the covariance matrix is

$$\mathbf{C}_x = \frac{1}{6} \begin{pmatrix} 4 + 1 + 0 + 0 + 1 + 4 & 0 - 2 + 0 + 0 + 2 + 8 \\ 0 - 2 + 0 + 0 + 2 + 8 & 0 + 4 + 9 + 1 + 4 + 16 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 5/3 & 4/3 \\ 4/3 & 5/3 \end{pmatrix}$$

Then we calculate the eigenvalues and eigenvectors of  $\mathbf{C}_x$ :

$$\mathbf{C}_x \mathbf{e}_i = \lambda_i \mathbf{e}_i \quad \Leftrightarrow \quad \begin{pmatrix} 5/3 & 4/3 \\ 4/3 & 5/3 \end{pmatrix} \begin{pmatrix} \mathbf{e}_{i1} \\ \mathbf{e}_{i2} \end{pmatrix} = \lambda_i \begin{pmatrix} \mathbf{e}_{i1} \\ \mathbf{e}_{i2} \end{pmatrix}$$

The solution is obtained when  $|\mathbf{C}_x - \lambda \mathbf{I}| = 0$ ,

$$\begin{vmatrix} 5/3 - \lambda & 4/3 \\ 4/3 & 5/3 - \lambda \end{vmatrix} = \left(\frac{5}{3}\right)^2 - 2 \cdot \frac{5}{3} \cdot \lambda + \lambda^2 - \left(\frac{4}{3}\right)^2 = \lambda^2 - \frac{10}{3} \lambda + 1 = 0 \Rightarrow \lambda_1 = 3 \text{ and } \lambda_2 = 1/3.$$

Next we calculate the eigenvectors

$$\Rightarrow \mathbf{e}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

The transform matrix  $\mathbf{A}$  is thus

$$\mathbf{A} = \begin{pmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

a) Transform 2-dim  $\Rightarrow$  1-dim: the transform matrix is the eigenvector that corresponds to the largest eigenvalue:

$$\mathbf{A} = \mathbf{e}_1^T = (1/\sqrt{2} \quad 1/\sqrt{2})$$

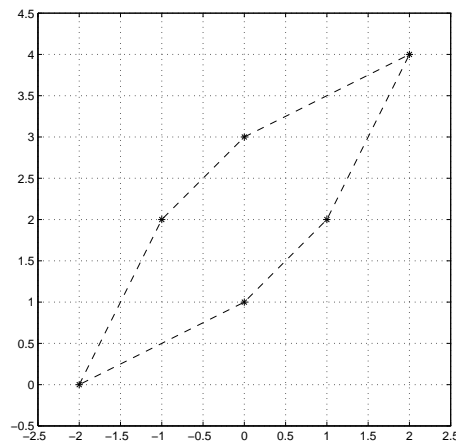
The average error is  $R = \lambda_2 = 1/3$ .

The transformed points are:

$$\mathbf{y}_1 = \mathbf{A}(\mathbf{x} - \mathbf{m}_x) = \mathbf{e}_1^T(\mathbf{x}_1 - \mathbf{m}_x) = (1/\sqrt{2} \quad 1/\sqrt{2}) \begin{pmatrix} -2 - 0 \\ 0 - 2 \end{pmatrix} = -2\sqrt{2}$$

$$\mathbf{y}_2 = -\sqrt{2}/2, \mathbf{y}_3 = \sqrt{2}/2, \mathbf{y}_4 = -\sqrt{2}/2, \mathbf{y}_5 = \sqrt{2}/2, \mathbf{y}_6 = 2\sqrt{2}.$$

b) The six sample points and the region spanned by them is shown below.



The elongatedness of the region can be obtained straight from the eigenvalues  $\lambda_1$  and  $\lambda_2$ . Since they correspond to variances of the sample points along the principal components, the elongatedness is obtained from their ratio:

$$E = \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}} = \sqrt{\frac{3}{1/3}} = 3.$$

So the elongatedness is now 3:1. (The square roots are needed to obtain the standard deviations from the variances.)

**T-61.5070 COMPUTER VISION, Exercise 10/08**

**Motivation**

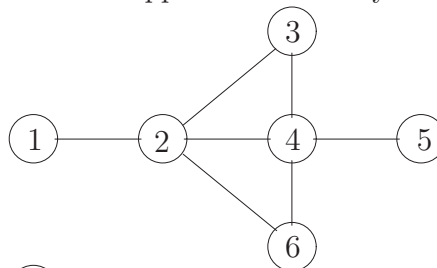
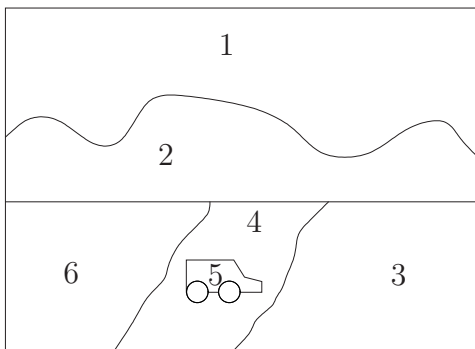
The purpose of this exercise is to be acquainted with both image understanding and some applications on image analysis.

1. One searches similarities between the relation graphs  $A$  and  $B$  based on cliques in the association graph.

$A$ : objects  $u, v, w, x, y, z$ ,  
 relations  $P(u), P(w), P(y), R(v), R(x), R(z), F(u, v), F(v, w), F(w, x), F(x, y), F(y, z), F(z, u)$

$B$ : objects  $a, b, c, d, e, f$   
 relations  $P(a), P(b), P(d), Q(e), Q(f), R(c), F(b, c), F(c, d), F(d, e), F(e, f), F(f, a)$

- (a) Construct the graphs corresponding to the relation structures  $A$  and  $B$ . Identify the nodes and the arcs.
  - (b) Construct the association graph corresponding to the relation structures  $A$  ja  $B$ .
  - (c) Search the largest cliques in the association graph visually. (There are only three of them.)
2. (a) Draw a graph representing a telephone equipped with a telephone dial.
  - (b) Introduce into your graph a telephone equipped with both a dial and keys.
  - (c) Introduce the telephone into an office.
3. Consider the given segmented image and the adjacency graph. Do a scene labelling by using the discrete relaxation method. Consider the application of unary constraints.



- ① Highest
- ⑤ Moving

## T-61.5070 COMPUTER VISION, Exercise 10/08

### 1.

Measures for graph similarity are shortly explained in Sec. 7.5.2/9.5.2, pp. 328/423.

a) The relation structures  $A$  and  $B$  are represented as graphs in Fig. 1. The unary relations (i.e., properties)  $P$ ,  $R$ , and  $Q$  are represented as circles, squares, and triangles, respectively. The binary relations (i.e., between two nodes) are represented as arrows, for example  $F(u, v)$  as an arrow from  $u$  to  $v$ .

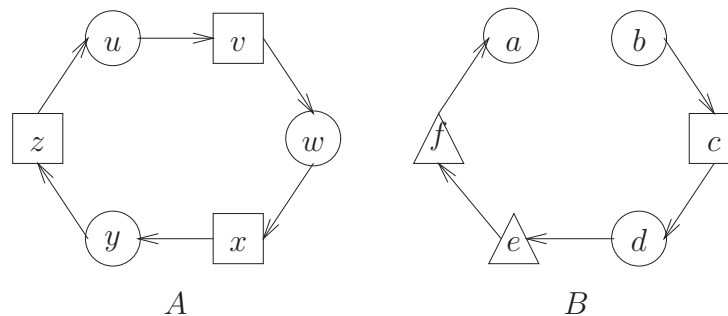


Figure 1: Graphs corresponding to the relation structures  $A$  and  $B$ .

b) Ballard and Brown defined the association graph in their book *Computer vision* (1982, p. 366) as follows:

The *association graph* defined in this section is an auxiliary data structure produced from two relational structures to be matched. [...]

Let a relational structure be a set of elements  $V$ , a set of properties (or more simply unary predicates)  $P$  defined over the elements, and a set of binary relations (or binary predicates)  $R$  defined over pairs of the elements. [...]

Given two structures defined by  $(V_1, P, R)$  and  $(V_2, P, R)$ , say that “similar” and “compatible” actually mean “the same”. Then we construct an association graph  $G$  as follows [...]. For each  $v_1$  in  $V_1$  and  $v_2$  in  $V_2$ , construct a node of  $G$  labeled  $(v_1, v_2)$  if  $v_1$  and  $v_2$  have the same properties [ $p(v_1)$  iff  $p(v_2)$  for each  $p$  in  $P$ ]. [...] Now connect two nodes  $(v_1, v_2)$  and  $(v'_1, v'_2)$  of  $G$  if they represent *compatible* assignments according to  $R$ , that is, if the pairs satisfy the same binary predicates [ $r(v_1, v'_1)$  iff  $r(v_2, v'_2)$  for each  $r$  in  $R$ ].

A match between  $(V_1, P, R)$  and  $(V_2, P, R)$  [...] is just a set of assignments that are mutually compatible. The “best match” could well be taken to be the largest set of assignments (node correspondences) that were all mutually compatible under the relations. But this in the association graph  $G$  is just the largest totally connected (completely mutually compatible) set of nodes. It is a *clique*. A clique to which no new nodes may be added without destroying the clique properties is a *maximal clique*. In this formulation of matching, larger cliques are taken to indicate better matches, since they account for more nodes. Thus the best matches are determined by the largest maximal cliques in the association graph.

The association graph corresponding to the relation structures  $A$  and  $B$  is depicted in Fig. 2. Note that the definition of clique above differs somewhat from the normal one, where all nodes should be connected to all other nodes within the clique. In this algorithm we only require the nodes be connected through other nodes.

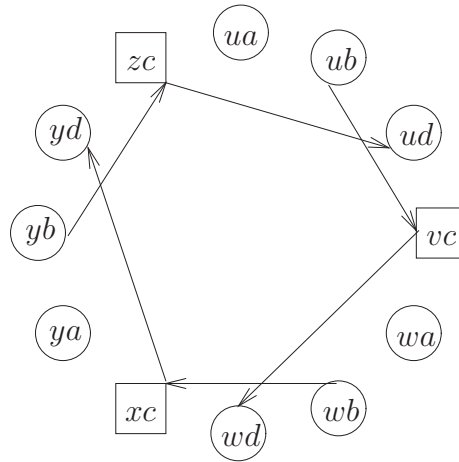


Figure 2: Association graph corresponding to the relation structures  $A$  and  $B$ .

c) There are three maximal cliques of the same size:  $\{ub, vc, wd\}$ ,  $\{wb, xc, yd\}$ , and  $\{yb, zc, ud\}$ .

**2.**

Semantic nets are explained in Sec. 7.1/9.1, pp. 294–295/384–386.

a) A semantic net for a simple dial telephone is depicted in Fig. 3

b) The network in Fig. 4 describes both rotary dial and pushbutton dial telephones. The node 'dial' is a generic concept and it has two instances, 'rotary' and 'pushbutton' which is expressed with the 'instance-of' relation.

c) The network in Fig. 5 depicts an office set with the elements table, window, drawer, and phone.

**3.**

Discrete relaxation is explained in Sec. 8.5.1/10.7.1, pp. 398–400/498–500. A set of binary constraints, adjacency relations, may be developed. Allowable labels on adjacent regions are:

- car is adjacent to road,  $a(C, R)$
- road is adjacent to grass,  $a(R, G)$
- grass is adjacent to trees,  $a(G, T)$
- road is adjacent to trees,  $a(R, T)$
- sky is adjacent to trees,  $a(S, T)$

A set of unary constraints or properties may be defined as

- sky is the highest region,  $h(S)$
- car is moving,  $m(C)$

The sample image was segmented into six regions (1–6) and their adjacency was represented as a graph. In the segmented image region 1 is the highest region and region 5 is moving. All possible labels are assigned to each region; the second column in Table 1. Considering the unary

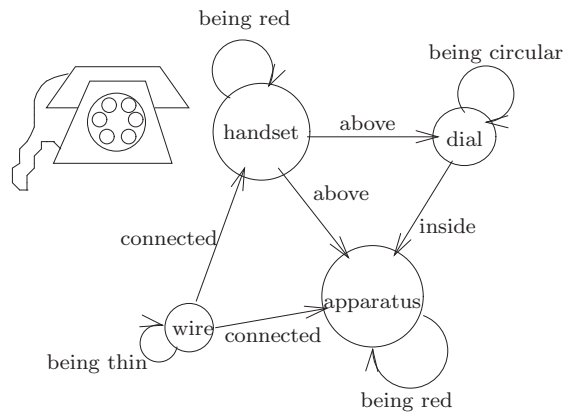


Figure 3: Semantic net for a simple dial telephone

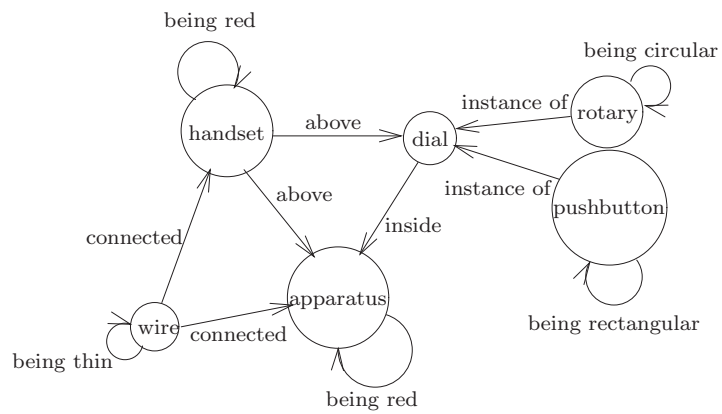


Figure 4: The network for telephones with either a rotary or a pushbutton dial.

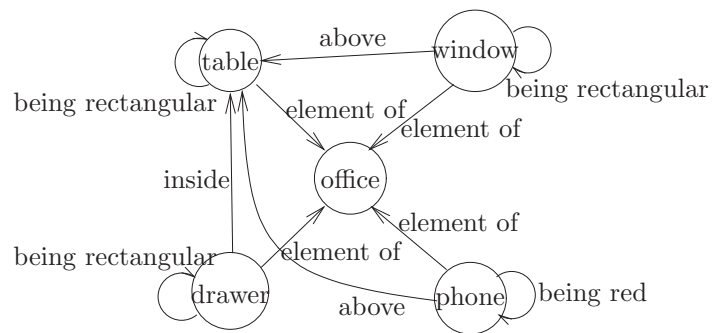


Figure 5: Office set.

constraints, locally inconsistent labels are removed; the third column in Table 1. Region 1 is labeled as 'sky' and region 5 is labeled as 'car'. The labels 'sky' and 'car' are removed from the other regions. Inconsistent labels are then deleted from the regions considering the binary relations; the fourth and fifth columns in Table 1.

1	STGRC	S	S	S
2	STGRC	TGR	T	T
3	STGRC	TGR	GR	G
4	STGRC	TGR	R	R
5	STGRC	C	C	C
6	STGRC	TGR	G	G

Table 1: Label removals in discrete relaxation.

Region 2 is chosen for updating. It is adjacent to region 'sky'. Therefore, according to the binary constraints, it cannot be labeled otherwise as 'trees'.

Region 3 is chosen for updating. It is adjacent to regions labeled as 'trees' (region 2) and as 'trees, grass, road' (region 4). It is labeled as 'grass, road'.

Region 4 is chosen for updating. It is adjacent to region labeled as 'car'. It is labeled as 'road'.

Region 6 is chosen for updating. It is adjacent to regions 'trees' (region 2) and 'road' (region 4).

'Grass' is the only possible region adjacent to both 'trees' and 'road'.

Region 3 is chosen for updating. It is adjacent to regions labeled as 'trees' (region 2) and 'road' (region 4). It is labeled as 'grass'.

If either of the unary constraints is removed then several solutions are possible for labeling, i.e., the problem becomes underconstrained. For instance, using the single unary constraint 'highest' results in four other solutions in addition to the "correct solution", Table 2.

1	S	S	S	S	S
2	T	T	T	T	T
3	G	R	R	G	G
4	R	G	G	R	R
5	C	R	T	G	T
6	G	R	R	G	G

Table 2: Possible labelings using single unary constraint highest.

**T-61.5070 COMPUTER VISION, Exercise 11/08****Motivation**

The purpose of this exercise is to be acquainted with motion analysis.

1. An object moves in 3D space with varying velocity. The motion of the object is recorded. Is it possible to determine FOE from the image sequence?
2. Some motion field detection approaches are dealt with on pages 682–685/753–757 in the textbook.
  - (a) Consider a segmented ternary image. Suggest an approach that is capable to determine the velocity of the motion field.
  - (b) What can be said about the suggested method when the brightness in the background is more than  $f_p$  or less than  $f_p$ ?
  - (c) How does nonhomogeneities in the background influence on the results?
3. Optical flow can be estimated by the following formula (Horn & Schunck 1980):

Initialize  $u^k$  ja  $v^k$  to zero,  $k = 0$

Iterate until the desired error criteria is fulfilled

$$\begin{aligned}u^k &= \bar{u}^{k-1} - f_x \frac{P}{D}, \\v^k &= \bar{v}^{k-1} - f_y \frac{P}{D}.\end{aligned}$$

Give an explanation to the formula.

## T-61.5070 COMPUTER VISION, Exercise 11/08

1.

Focus of expansion (FOE, *F. laajentumis piste*) is described in Sec. 15.2.4/16.2.4, pp. 693–696/764–767. For an object moving with constant velocity, all image plane flow vectors intersect at FOE. A change in motion direction results in change of velocity, and a change in FOE location. If the movement of object points cannot be described by simple constant velocity translation, e.g. when the object is rotating, no FOE may be determined. In general, a FOE does not exist for objects moving with nonconstant 3-D velocities. For the special case when the object speed varies while the the velocity direction stays constant the FOE can be determined, however.

2.

A moving rectangle has been chosen as an example of image motion for this exercise, Fig. 1.

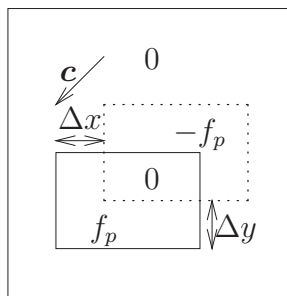


Figure 1: Ternary image example.

Gray-scale values are  $f_p$  for the object and 0 for the background. New images of the moving rectangle are taken at a constant interval  $T$ . The difference image between two subsequent frames has four regions with three intensities:

1. A region of intensity  $-f_p$ , because of the uncovering of the background over the time interval;
2. A region of intensity  $f_p$ , because of covering the background over the time interval;
3. A region between the above two of intensity 0, because of the presence of the object in this region in both images; and
4. A region of intensity 0, because of the unchanged background in both images.

The difference image thus has three different possible values:  $-f_p$ , 0 and  $f_p$ , hence the name ternary image.

a) Determine changes  $\Delta x$  and  $\Delta y$  in the direction from the region with intensity of  $-f_p$  to the region with intensity  $f_p$ . The velocity is  $\mathbf{c} = [\frac{\Delta x}{T} \ \frac{\Delta y}{T}]^T$ .

b) The background intensity is thresholded to value  $f_0$  instead of 0. If

$$f_0 > f_p \Rightarrow f_p - f_0 < 0,$$

then the direction of velocity is from the region with positive intensity values to the region with negative values. If

$$f_0 < f_p \Rightarrow f_p - f_0 > 0,$$

then the situation is the same as in section a).

c) If the background intensity  $f_0(x)$  varies being sometimes identical to, or higher or lower than the object intensity, then the differentiation between object and background becomes difficult. In that case a proper segmentation cannot be obtained with simple thresholding; its use in this case may lead to an inaccurate and erroneous local velocity.

### 3.

Computation of optical flow is represented in Sec. 15.2.1/16.2.1, pp. 686–689/757–760. Algorithm 15.1/16.1, p. 687/759, determines optical flow from a pair of consecutive images  $f(x, y, t)$  and  $f(x, y, t + 1)$ . The velocity components  $u(x, y)$  and  $v(x, y)$  in the  $x$  and  $y$  directions, respectively, are computed iteratively according to the equations:

$$\begin{aligned} u^k &= \bar{u}^{k-1} - f_x \frac{P}{D} \\ v^k &= \bar{v}^{k-1} - f_y \frac{P}{D}, \end{aligned} \quad (1)$$

where  $\bar{u}$  and  $\bar{v}$  are averages of the velocity components in a given neighborhood and  $P$  and  $D$  are defined as

$$P = f_x \bar{u} + f_y \bar{v}, \quad (2)$$

$$D = \lambda^2 + f_x^2 + f_y^2. \quad (3)$$

This Gauss-Seidel iteration finds the values of  $u$  and  $v$  that minimize the function

$$E^2(x, y) = (f_x u + f_y v + f_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2). \quad (4)$$

The values of  $u$  and  $v$  providing the minimum of Eq. 4 are the solution for the velocity vector problem (see the book, pp. 686–687/757–759).

The values of  $f_x$ ,  $f_y$ , and  $f_t$  can be estimated from the image pair, for instance,

$$\begin{aligned} f_x(x, y) &= f(x + 1, y, t) - f(x, y, t), \\ f_y(x, y) &= f(x, y + 1, t) - f(x, y, t), \text{ and} \\ f_t(x, y) &= f(x, y, t + 1) - f(x, y, t). \end{aligned} \quad (5)$$

The coefficient  $\lambda$  is a Lagrange multiplier used in constrained nonlinear programming problems<sup>2</sup>.

The Eqs. (15.9/16.9) in the textbook yield two equations that may be solved for  $\lambda^2$  in Eq. 3:

$$\begin{aligned} \lambda^2 &= \frac{f_x f_y v + f_x^2 u + f_x f_t}{\bar{u} - u} \\ &= \frac{f_x f_y u + f_y^2 v + f_y f_t}{\bar{v} - v}. \end{aligned} \quad (6)$$

In Eqs. 1, the averages  $\bar{u}$  and  $\bar{v}$  have a constant effect on the new values  $u^k$  and  $v^k$ . The effect of the image gradient vanishes when the gradient is perpendicular to the average velocity, i.e.,  $P = \nabla \mathbf{f} \cdot \bar{\mathbf{c}} = 0$ . Otherwise, some fraction of the image gradient,  $-\frac{P}{D} \nabla \mathbf{f}$ , is added to the average velocity, see Fig. 2.

If the neighborhood were not taken into account, then optical flow would always be determined in the direction of the local image gradient. The method of Horn and Schunck makes a trade-off between velocity in the direction of the local image gradient at  $(x, y)$  and the average velocity in its neighborhood. The method is based on matching the direction of the image gradient to the average velocity (in  $P$ ), and on the magnitude of the image gradient (in  $D$ ), for instance.

<sup>2</sup>Minimization of a scalar function  $F(\mathbf{x})$  subject to constraints  $h_i(\mathbf{x}) \geq 0$  leads to minimization of the Lagrange function  $L(\mathbf{x}, \boldsymbol{\lambda}) = F(\mathbf{x}) + \sum_{i=1}^m \lambda_i h_i(\mathbf{x})$ .

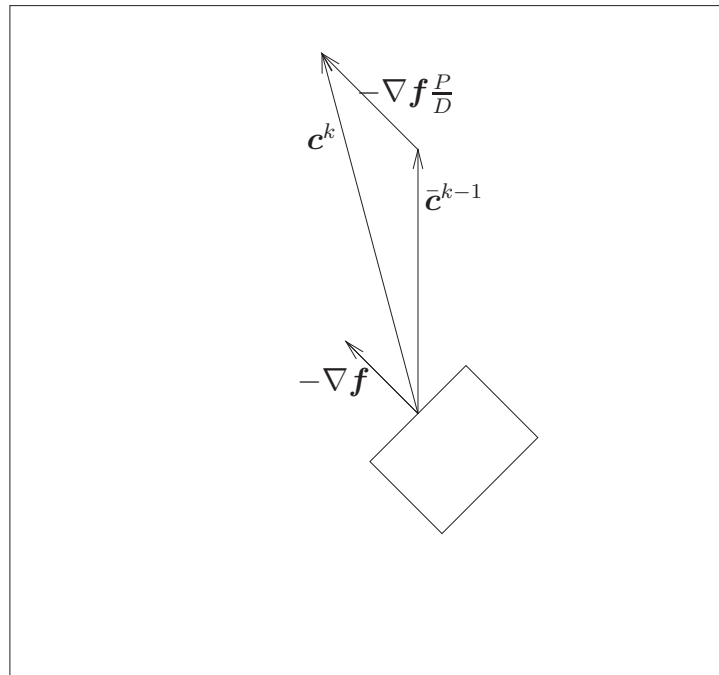


Figure 2: Local velocity is updated according to the average velocity,  $\bar{c}^{k-1}$ , in a given neighborhood, which is added to some fraction of the image gradient,  $-\nabla f \frac{P}{D}$ .

**Notice** The suggested initialization  $\mathbf{c} = 0$  is not good because  $\bar{c}$  and  $P$  are then zero and any new velocity component values are also zero. A better initialization is provided in the textbook by Eq. (15.7/16.7), according to which the component of velocity  $\mathbf{c}$  in the direction of image gradient  $\nabla \mathbf{f}$  is  $-f_t$  and therefore  $u$  and  $v$  may be initialized to

$$\begin{aligned}
 u^0 &= \frac{-f_t f_x}{\sqrt{f_x^2 + f_y^2}}, \\
 v^0 &= \frac{-f_t f_y}{\sqrt{f_x^2 + f_y^2}}.
 \end{aligned}
 \tag{7}$$