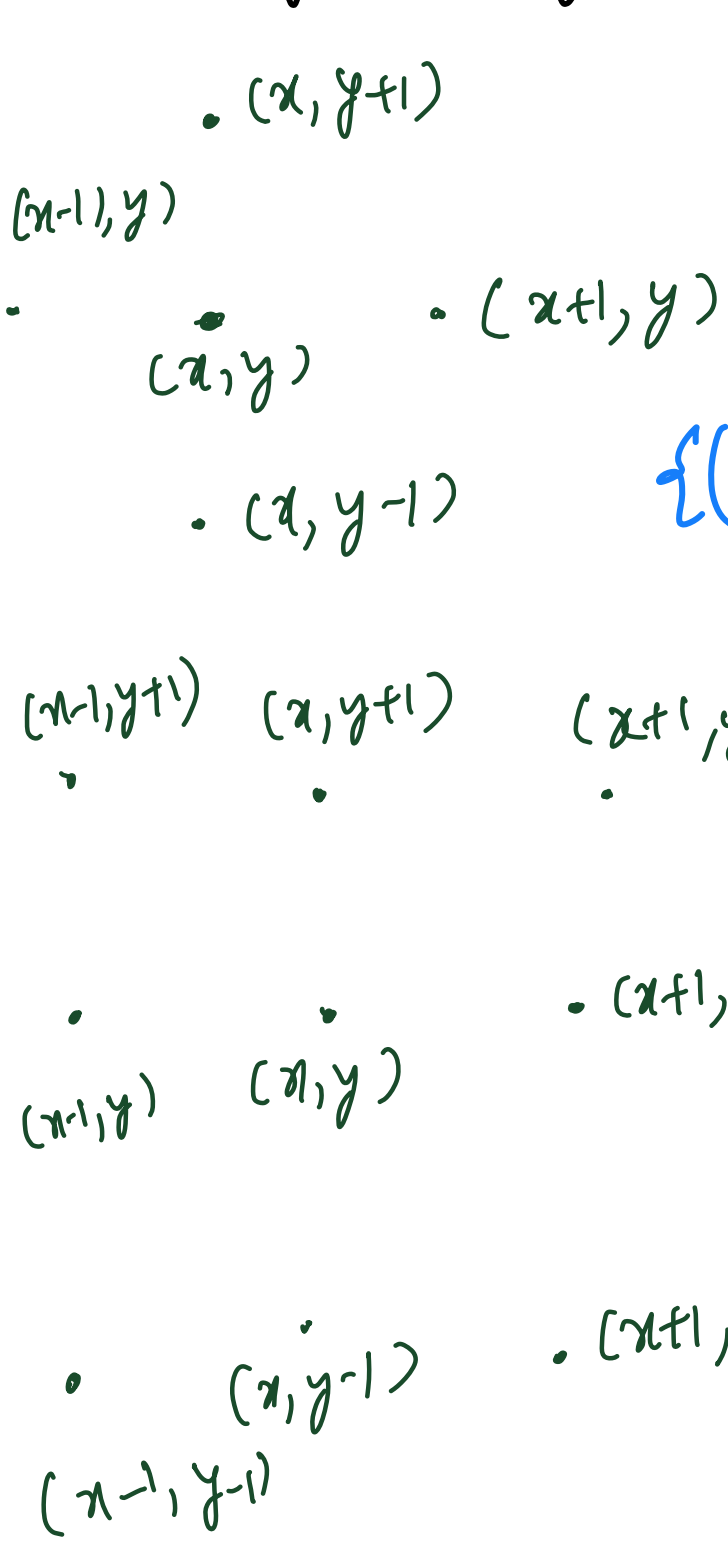
Concise CV

CHAPTER-3

Chapter 3: Image Analysis

Basic Image Topology

4-8 Adjacency of Binary Image



4-Adjacency of of pixel $p(x,y)$ is

$A_4(p) = p + A_4 =$

$\{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$

8-Adjacency of pixel $p(x,y)$ is $A_8(p)$

$p + A_8 =$

$\{(x+1, y), (x+1, y+1), (x+1, y-1), (x, y-1), (x, y+1), (x-1, y+1), (x-1, y), (x-1, y-1)\}$

→ As is the kind of edge and corner adjacency set

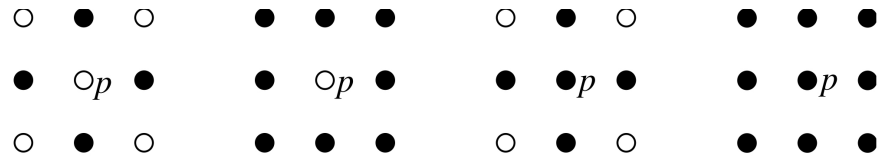


Fig. 3.2 *Left:* 4-adjacency set and 8-adjacency set of p. *Right:* 4-neighbourhood and 8-neighbourhood of p

Pixel Neighbourhoods A *neighbourhood* of a pixel p contains the pixel p itself and some adjacent pixels. For example, the *4-neighbourhood* of p equals $A_4(p) \cup \{p\}$, and the *8-neighbourhood* of p equals $A_8(p) \cup \{p\}$. See Fig. 3.2.

Insert 3.2 (Adjacency, Connectedness, and Planarity in Graph Theory) *An (undirected)* graph $G = [N, E]$ *is defined by a set* N *of* nodes *and a set* E *of* edges; *each edge connects two nodes. The graph* G *is* finite *if* N *is finite.*

Two nodes are adjacent *if there is an edge between them. A* path *is a sequence of nodes, where each node in the sequence is adjacent to its predecessor.*

A set $S \subseteq N$ *of nodes is* connected *iff there is a path in* S *from any node in* S *to any node in* S. *Maximal connected subsets of a graph are called* components.

A planar graph *can be drawn on the plane in such a way that its edges intersect only at their endpoints (i.e. nodes). Let* α_1 *be the number of edges, and* α_0 *be the number of nodes of a graph. For a planar graph with* $\alpha_0 \geq 3$, *we have that* $\alpha_1 \leq 3\alpha_0 - 6$; *if there are no cycles of length* 3 *in the graph, then it is even* $\alpha_1 \leq 2\alpha_0 - 4$.

Euler's formula *states that for a finite planar and connected graph,* $\alpha_2 - \alpha_1 + \alpha_0 = 2$, *where* α_2 *denotes the number of faces of the planar graph.*

Pixel Connectness

Let $S \subseteq \mathbb{Z}^2$ then if:

1) A pixel is connected to itself

2) Adjacent pixel in S are connected

3) If pixel $p \in S$ and pixel $q \in S$ is adjacent to the pixel $r \in S$, then p is also connected to r (in S)

→ Depending on choosen adjacency, we have either 4-connectedness or 8-connectedness of subset of R

Region :- Maximal set of connected pixel is called regions or components

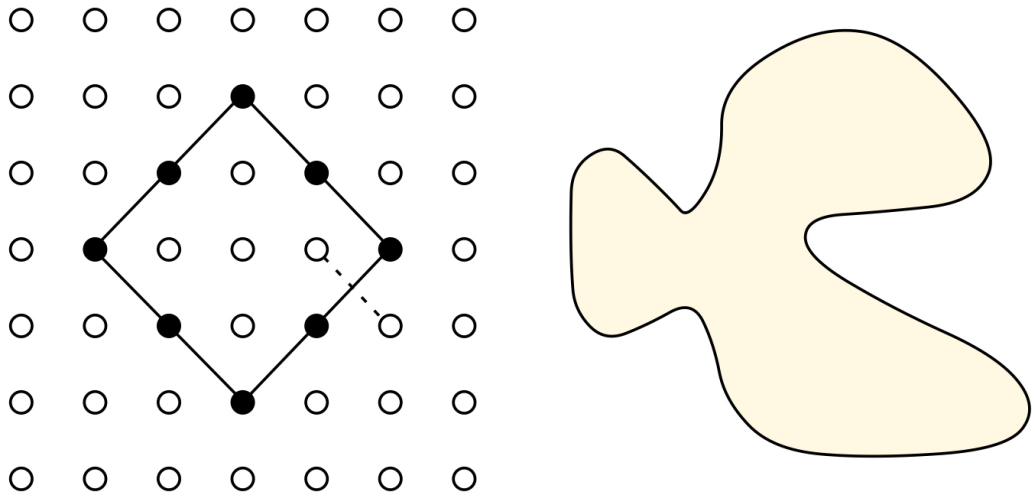


Fig. 3.3 *Left*: Assume 4-adjacency: The disconnected black pixels separate a connected "inner region" from a connected "outer region". Assume 8-adjacency: The black pixels are connected (as illustrated by the inserted edges), but all the white pixels remain connected (see the dashed edge as an example). *Right*: A simple curve in the Euclidean plane always separates interior (the shaded region) from exterior

Dual Adjacencies in Binary Images Figure 3.3 illustrates the consequences when deciding for one particular type of adjacency by drawing a comparison with the geometry in the Euclidean plane $\mathbb{R}^2$. $\mathbb{R}$ is the set of all real numbers. A *simple curve*, also known as a *Jordan curve*, always separates an inner region, called the *interior*, from an outer region, called the *exterior*. This appears to be obvious, in

→ Mathematical proof of this property is known as Jordan-Brower theorem

Topological Sound Pixel Adjacency

Assume black > grey > white. So Corner is black if any pixel is black, corner is grey if no pixel

is black and only grey or grey and white pixel present. Else corner is white

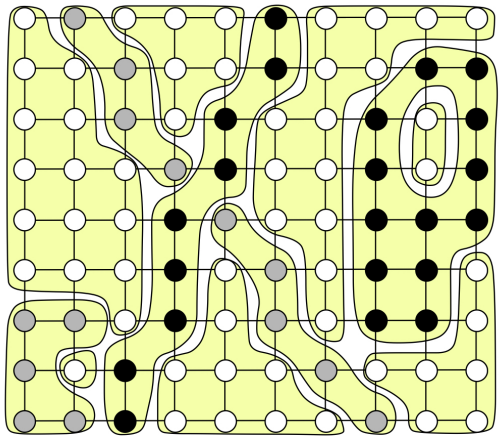


Fig. 3.7 Components for "black > grey > white"

2-black connected components 3-grey and 7-white

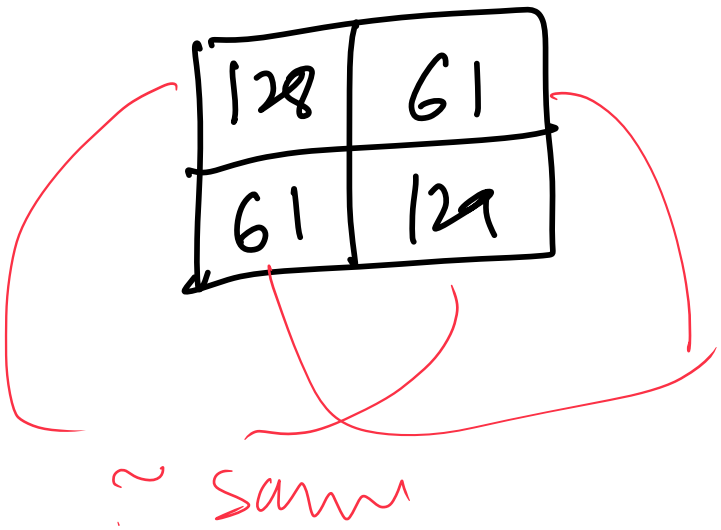
→ The order of importance define the key for adjacency thus, we call it R-adjacency.

→ For scalar image order of grey level

→ For vector valued image order of magnitude followed by lexographic values in

case of equal magnitude

Flip-Flop Case



Booder Tracing

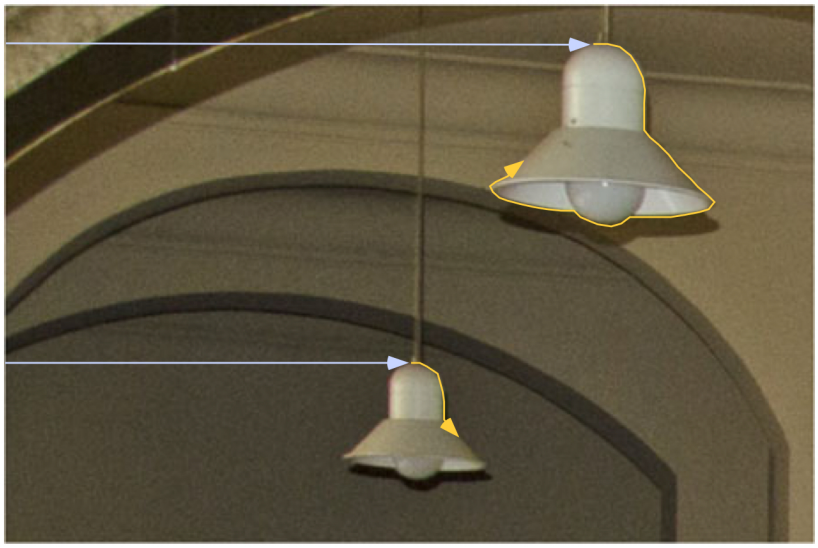


Fig. 3.10 Illustration of two scanlines that arrive for the first time (assuming a standard scan: *top–down*, *left* to *right*) at objects of interest (lights). At this moment a tracing procedure starts for going around on the border of the object

→ when arriving at scan line of the object we like to trace booder

such that either the object lies either on right or on left

- The adjacency might be 4-, 8- or 4 adjacency
- At every pixel p, we have local circular order $\xi(p) \doteq \langle q_1 \cdots q_n \rangle$ which list all adjacent pixel in $A(p)$ only once

Assume that we arrive at $p_{i+1} \in A(p_i)$. Let q_1 be the pixel next to pixel p_i in the local circular order of p_{i+1}. We test whether q_1 is in the object; if "yes", then we have $p_{i+2} = q_1$ and continue at p_{i+2}; if "not", then we test the next pixel q_2 in the local circular order $\xi(p_{i+1})$ of p_{i+1}, and so forth.

Voss Algorithm

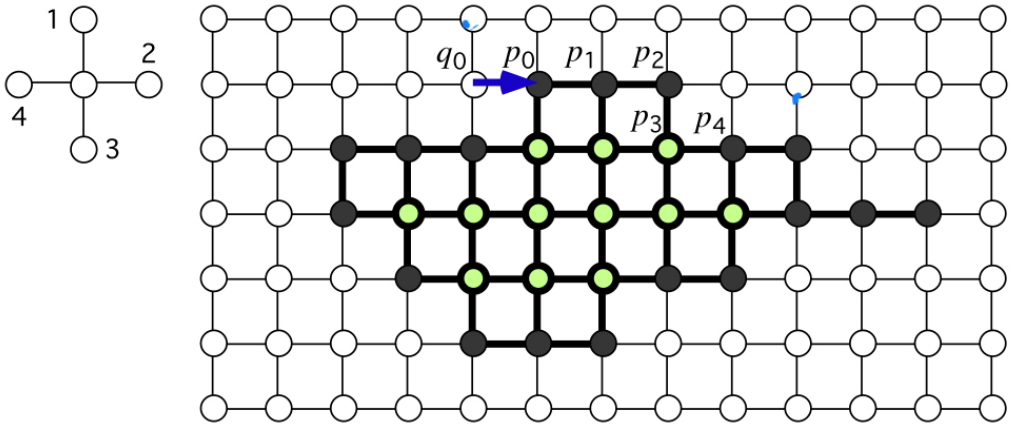


Fig. 3.11 *Left*: Used local circular order. *Right*: Arrival at an object when going from q_0 to p_0

1: Let $(q_0, p_0) = (q, p)$, $i = 0$, and $k = 1$;
2: Let q_1 be the pixel which follows q_0 in $\xi(p_0)$;
3: **while** $(q_k, p_i) \neq (q_0, p_0)$ **do**
4: **while** q_k in the object **do**
5: Let $i := i + 1$ and $p_i := q_k$;
6: Let q_1 be the pixel which follows p_{i-1} in $\xi(p_i)$ and $k = 1$;
7: **end while**
8: Let $k = k + 1$ and go to pixel q_k in $\xi(p_i)$;
9: **end while**
10: The calculated border cycle is $\langle p_0, p_1, \ldots, p_i \rangle$;

Fig. 3.12 Voss algorithm

Area

Area of $\Pi(p, q, r)$ where $p = (x_1, y_1)$ $q = (x_2, y_2)$ and $r = (x_3, y_3)$ have area:

$$A(T) = \frac{1}{2} |D(p, q, r)|$$

where $D(p,q,r)$ is

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1(y_2 - y_3) - y_1(x_2 - x_3) + 1(x_2 y_3 - x_3 y_2)$$

$$x_1 y_2 - x_1 y_3 - y_1 x_2 + y_1 x_3 + x_2 y_3 - x_3 y_2$$

The area of a simple polygon $\Pi = \langle p_1, p_2, \ldots, p_n \rangle$ in the Euclidean plane, with $p_i = (x_i, y_i)$ for $i = 1, 2, \ldots, n$, is equal to

$$\mathscr{A}(\Pi) = \frac{1}{2}\left|\sum_{i=1}^{n} x_i (y_{i+1} - y_{i-1})\right| \tag{3.5}$$

for $y_0 = y_n$ and $y_{n+1} = y_1$. In general, the area of a compact set R in $\mathbb{R}^2$ equals

$$\mathscr{A}(R) = \int_R \mathrm{d}x\,\mathrm{d}y \tag{3.6}$$

How to measure area ?

Figure 3.14 illustrates an experiment. We generate a simple polygon in a grid of size 512×512 and subsample it in images of reduced resolution. The original polygon Π has the area 102,742.5 and perimeter 4,040.7966... in the 512×512 grid.

For the *perimeter* of the generated polygons, we count the number of cell edges on the frontier of the polygon times the length of an edge for the given image resolution. For the 512×512 image, we assume the edge length to be 1, for the 128×128 image, the edge length to be 4, and so forth.

For the *area* of the generated polygons, we count the number of pixels (i.e. grid cells) in the polygon times the square of the edge length.

The *relative deviation* is the absolute difference between the property values for the subsampled polygon and original polygon Π, divided by the property value for Π.

Figure 3.15 summarizes the errors of those measurements by showing the relative deviations. It clearly shows that the measured perimeter for the subsampled polygons is not converging towards the true value; the relative deviations are even increasing!

Regarding the measurement of the area of a region in an image, since the times of Gauss, it is known that the number of grid points in a convex set S estimates the area of S accurately. Thus, not surprisingly, the measured area shows the convergence towards the true area as the image size increases.

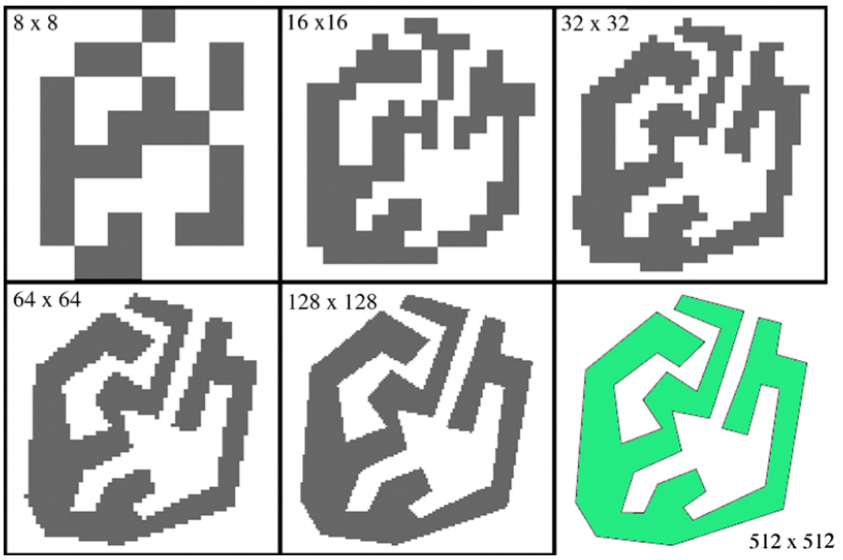


Fig. 3.14 Different digitizations of a simple polygon Π using grids of size 8×8 to 128×128; the original polygon was drawn on a grid of resolution 512×512. All images are shown in the grid cell model

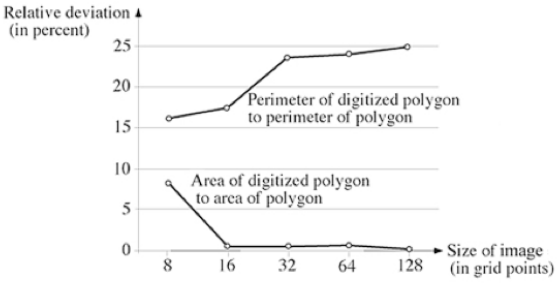
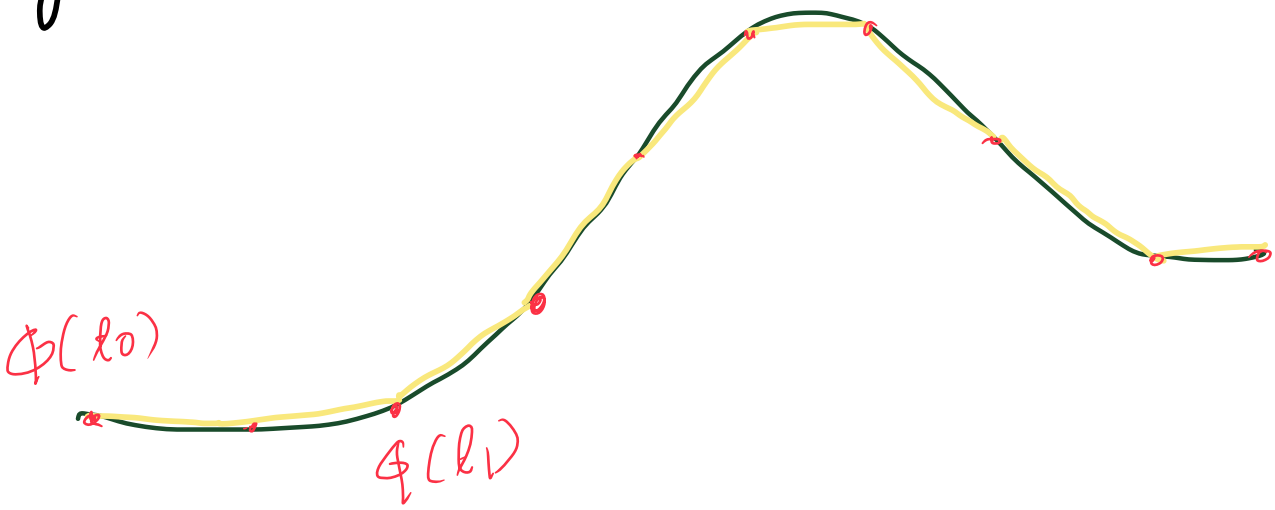


Fig. 3.15 Relative deviations of the area and perimeter of subsampled polygons relatively to the true value in the 512×512 grid

Observation 3.4 *The number of grid points in a region is a reliable estimator for the area of the shown object.*

Length



Assume that we have a parameterized one-to-one representation $\phi(t)$ of an arc γ, starting with $\phi(c)$ and ending at $\phi(d)$ for $c < d$. Values $t_0 = c < t_1 < \cdots < t_n = d$ define a polygonal approximation of this arc; see Fig. 3.16.

A polygonal approximation has a defined length (i.e. the sum of lengths of all line segments on this polygonal path). The limits of lengths of such polygonal approximations, as n tends to infinity (i.e. as line segments become smaller and smaller), define the length of γ.

Insert 3.8 (Jordan Arcs) *The general mathematical definition of an arc is as follows*: A Jordan arc γ *is defined by a subinterval* $[c, d]$ *of a Jordan curve* (*or simple curve*)

$$\{(x, y) : \phi(t) = (x, y) \wedge a \le t \le b\}$$

with $a \le c < d \le b$, $\phi(t_1) \ne \phi(t_2)$ *for* $t_1 \ne t_2$, *except* $t_1 = a$ *and* $t_2 = b$.

A rectifiable Jordan arc γ *has a bounded* arc length *as follows*:

$$\mathscr{L}(\gamma) = \sup_{n \ge 1 \wedge c = t_0 < \cdots < t_n = d} \sum_{i=1}^{n} d_e\big(\phi(t_i), \phi(t_{i-1})\big) < \infty$$

See Fig. 3.16. *In* 1883, *Jordan proposed the following definition of a curve*:

$$\gamma = \{(x, y) : x = \alpha(t) \wedge y = \beta(t) \wedge a \le t \le b\}$$

G. Peano showed in 1890 *that this allows a curve that fills the whole unit square. The Peano curve is not differentiable at any point in* $[0, 1]$. *Thus, Jordan's* 1883 *definition is used for arc length calculation*:

$$\mathscr{L}(\gamma) = \int_a^b \sqrt{\left(\frac{d\alpha(t)}{dt}\right)^2 + \left(\frac{d\beta(t)}{dt}\right)^2}\, dt$$

(*assuming differentiable functions* α *and* β).

Staircase Effect

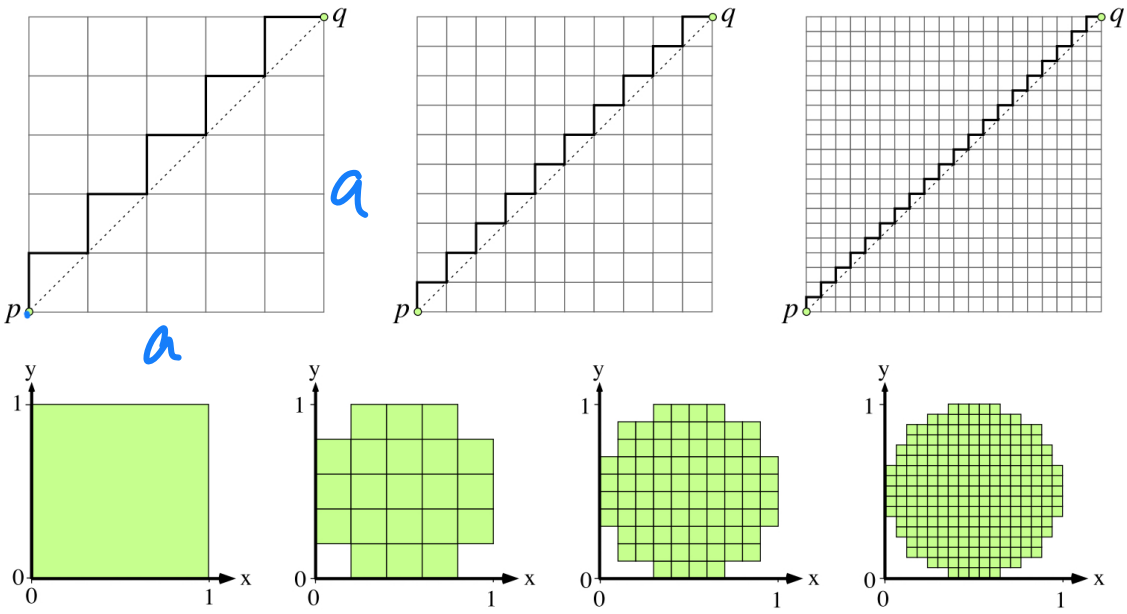


Fig. 3.17 *Top*: Approximations of the diagonal in a square by 4-paths for different grid resolutions. *Bottom*: Digitizations of a unit disk for different grid resolutions

In image 1, if we approximate the diagonal the diagonal is $a\sqrt{2}$ where a is the grid size. Using 4-path approximation length comes out to $2a$ irrespective of grid resolution taken. Similarly for the bottom image the fourier

boundary is always 4.

So due to 4-path approximation.

Error is

$$\frac{a\sqrt{2} - 2a}{a\sqrt{2}} \times 100 = 41.4\%$$

→ So this method is not recommended in length measurement in image analysis

8-path

Fig. 3.18 Approximation of line segment pq by 8-paths for different grid resolutions

In 8 path diagonal $\sqrt{2}$ x times grid and line parallel to axis is 1 x grid.

first image diagonal is $(3+\sqrt{2})/5$ and $(5+5\sqrt{2})/5$ for grid $1/2^n$

$n \geq 1$

→ So we understood the 8-path is similar to 4-path with only difference that here error is 7.9%.

This upper bound for magnitudes of errors might be acceptable in some applications. The use of weighted edges (including diagonal edges) for length estimation is certainly acceptable for low image resolution or relatively short digital arcs.

Polygonal Simplification of Borders What is a certified accurate way for measuring length? We go back to the scheme illustrated in Fig. 3.16. If segmenting an arc into maximum-length *digital straight segments* (DSS*s*), as illustrated in Fig. 3.19, then the sum of lengths of those straight segments converges to the true length of a digitized arc, provided that we have the budget to acquire equipment with finer and finer grid resolution.

Curvature

→ A Jordan curve is called smooth if it is continuously differentiable.

polygon is not smooth at singular points i.e vertices. Curvature is defined only at non-singular points of a curve.

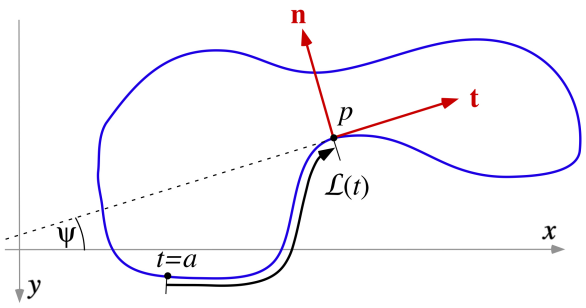


Fig. 3.20 The Frenet frame at $p = \gamma(t)$, also showing length $l = \mathcal{L}(t)$

Curvature as Rate of Change of Tangential Angle Assume an arc γ in the Euclidean plane that is a segment of a smooth Jordan curve. Thus, we have a tangent $\mathbf{t}(p)$ defined at any point p on γ. This tangent describes an angle ψ with the positive x-axis, called the *slope angle*. See Fig. 3.20.

Insert 3.10 (Frenet Frame and Frenet) *To define curvature it is convenient to use the* Frenet frame, *which is a pair of orthogonal coordinate axes (see Fig. 3.20) with origin at a point $p = \gamma(t)$ on the curve, named after the French mathematician J.F. Frenet* (1816–1900). *One axis is defined by the* tangent vector

$$\mathbf{t}(t) = \left[\cos\psi(t), \sin\psi(t)\right]^\top$$

where ψ is the slope angle *between the tangent and the positive x-axis. The other axis is defined by the* normal vector

$$\mathbf{n}(t) = [-\sin\psi(t), \cos\psi(t)[^\top$$

While p is sliding along γ, the angle ψ will change. The rate of change in ψ (with respect to the movement of p on γ) is one way to define *curvature* $\kappa_{tan}(p)$ along γ.

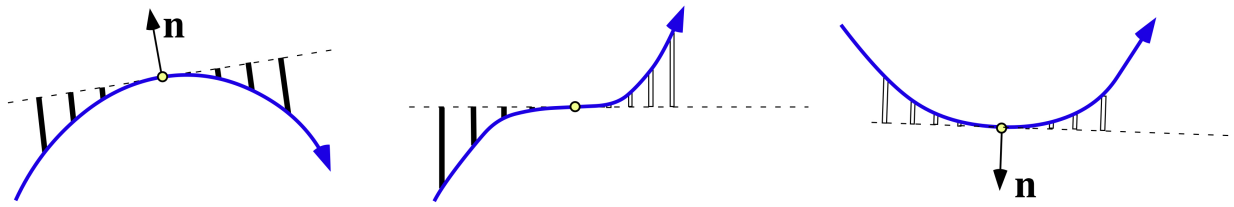


Fig. 3.21 Three tangents (*dashed lines*). The curvature is positive on the *left*, has a zero crossing in the *middle*, and is negative on the *right*

Let $l = \mathcal{L}(t)$ be the arc length b/w starting point $\gamma(a)$ and point general point $p = \gamma(t)$

Rate of parametrization of γ is given as

$$\kappa_{tan}(t) = \frac{\partial \psi(t)}{dt}$$

$$v(t) = \frac{\partial \mathcal{L}(t)}{dt}$$

at point $\gamma(t) = (x(t), y(t))$ of smooth Jordan curve γ

For $4\tan(t)$ – the rate of change can be +ve or -ve

If +ve at p → concave point

If -ve at p → Convex point

As Fig. 3.21 shows, the situation at p can be approximated by measuring the distances between γ and the tangent to γ at p along equidistant lines perpendicular to the tangent. In Fig. 3.21, positive distances are represented by bold line segments and negative distances by "hollow" line segments. The area between the curve and the tangent line can be approximated by summing these distances; it is positive on the left, negative on the right, and zero in the middle where the positive and negative distances cancel.

For example, assume that γ is a straight line. Then the tangent coincides with γ at any point p, and there is no rate of change at all, i.e. there is curvature zero for all points on γ. Assume that γ is a circle. Then we already know that there is a constant rate of change in slope angle; assuming uniform speed, it follows that this constant is the inverse of the radius of this circle.

Curvature as radius of osculating circle

The osculating circle at point p of γ is the largest circle tangent to γ at p on the concave side

Assume that osculating circle at radius r then γ has curvature

$$\kappa_{osc}(p) = \frac{1}{r} \text{ at } p$$

In case of straight line r is infinity and the curvature is 0.

It holds

$$\kappa_{osc}(p) = \kappa_{tan}(p)$$

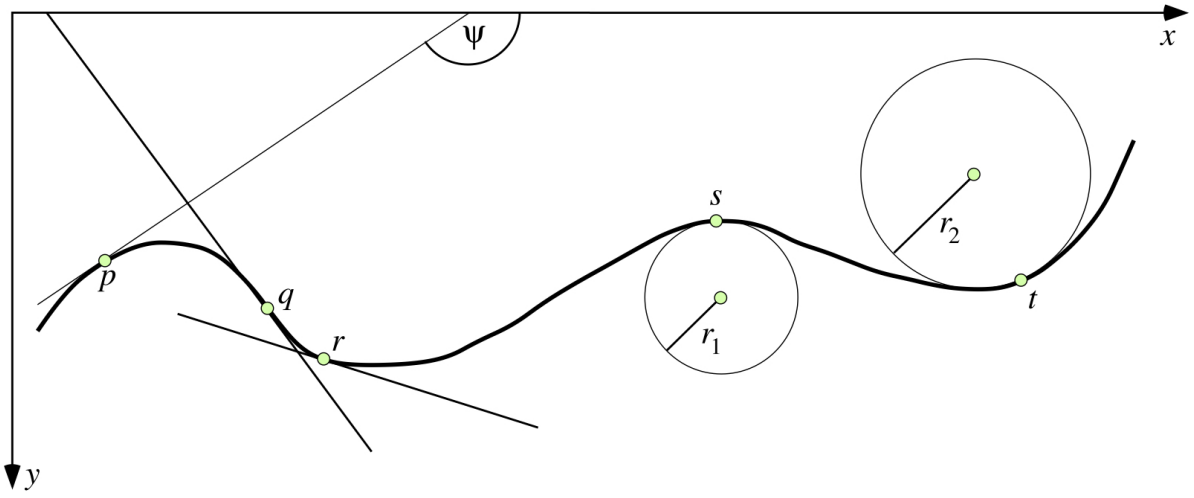


Fig. 3.22 Illustration of curvature defined either by rate of change (*left*) or radius of osculating circle (*right*). Points p, q, and r illustrate curvature by rate of change in the slope angle ψ. Assume that points move from *left* to *right*. Then p has a positive curvature, zero curvature at a point of inflection q, and negative curvature at r. Points s and t illustrate curvature by their osculating circles. The curvature at s equals $1/r_1$, and at t it is $1/r_2$

Curvature of a Parameterized Arc There is a third option for defining curvature in the Euclidean plane. Assume a parametric representation $\gamma(t) = (x(t), y(t))$, which is Jordan's first proposal for defining a curve. Then it follows that

$$\kappa_{tan}(t) = \frac{\dot{x}(t) \cdot \ddot{y}(t) - \dot{y}(t) \cdot \ddot{x}(t)}{[\dot{x}(t)^2 + \dot{y}(t)^2]^{1.5}} \tag{3.10}$$

where

$$\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}, \qquad \dot{y}(t) = \frac{\mathrm{d}y(t)}{\mathrm{d}t}, \qquad \ddot{x}(t) = \frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2}, \qquad \ddot{y}(t) = \frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2} \tag{3.11}$$

Algorithms for estimating curvature of digital curves in images are a subject of digital geometry. Typically, they follow the rate-of-change model, rather than the osculating circle model, and only a few attempt to digitize (3.10).

Distance Transform (By Gisella Klette)

The *distance transform* labels each object pixel (say, defined by $I(p) > 0$) with the Euclidean distance between its location and the nearest non-object pixel location (defined by $I(p) = 0$). For simplicity, we can say that the distance transform determines for *all* pixel locations $p \in \Omega$ the distance value

$$D(p) = \min_{q \in \Omega} \{ d_2(p, q) : I(q) = 0 \} \tag{3.15}$$

where $d_2(p, q_k)$ denotes the Euclidean distance. It follows that $D(p) = 0$ for all non-object pixels.

Maximal Circles in the Image Grid Let $S \subset \Omega$ be the set of all object pixel locations, and $B = \Omega \setminus S$ be the set of all non-object pixel locations. The distance transform satisfies the following properties:

1. $D(p)$ represents the radius of the largest disk centred at p and totally contained in S.
2. If there is only one non-object pixel location $q \in B$ with $D(p) = d_2(p, q)$, then there are two cases:
   (a) There exists a pixel location $p' \in S$ such that the disk centred at p' totally contains the disk centred at p, or
   (b) there exist pixel locations $p' \in S$ and $q' \in B$ such that $d_2(p, q) = d_2(p', q')$ and p is 4-adjacent to p'.
3. If there are two (or more) non-object pixel locations $q, q' \in B$ such that $D(p) = d_2(p, q) = d_2(p, q')$, then the disk centred at p is a maximal disk in S; the point p is called *symmetric* in this case.

In Case 2(b), the pixel locations p and p' are both centres of maximal discs, and thev are 4-adiacent to each other.

Distance and Row–Column Component Map The *distance map* is a 2D array of the same size as the original image that stores the results $D(p)$ at locations $p \in \Omega$.

Let a shortest distance $D(p)$ be defined by the distance $d_2(p, q)$ with $p = (x_p, y_p)$ and $q = (x_q, y_q)$. Then we have that

$$D(p) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \tag{3.16}$$

By knowing $\triangle x = x_p - x_q$ and $\triangle y = y_p - y_q$ we also know $D(p)$, but just the distance value $D(p)$ does not tell us the signed *row component* $\triangle x$ and the signed *column component* $\triangle y$. Thus, instead of the distance map, we might also be interested in the *row–column component map*: At $p \in \Omega$ we store the tuple $(\triangle x, \triangle y)$ that defines $D(p)$.

Square Euclidian distance Transform (SEDT)

- Refer book for understanding

Coocurence Matrix and Measures

Coccurence studies is the dissibution of values dependence upon the adjacent pixel. Such co-occurence result is represented by co-occurn matrix

Assume an input image I and an adjacency set A. For example, in case of 4-adjacency we have the adjacency set $A_4 = \{(0, 1), (1, 0), (0, -1), (-1, 0)\}$, defining $A_4(p) = A_4 + p$ for any pixel location p. As before, we denote by Ω the set of all $N_{cols} \times N_{rows}$ pixel locations. We define the $(G_{\max} + 1) \times (G_{\max} + 1)$ co-occurrence matrix $\mathbf{C}_I$ for image I and image values u and v in $\{0, 1, \ldots, G_{\max}\}$ as follows:

$$\mathbf{C}_I(u, v) = \sum_{p \in \Omega} \sum_{q \in A \wedge p+q \in \Omega} \begin{cases} 1 & \text{if } I(p) = u \text{ and } I(p+q) = v \\ 0 & \text{otherwise} \end{cases} \tag{3.33}$$

The example illustrates two general properties of those co-occurrence matrices:

1. Each element q in the adjacency set adds either $N_{cols} \cdot (N_{rows} - 1)$ or $(N_{cols} - 1) \cdot N_{rows}$ to the total sum of entries in the co-occurrence matrix, depending on whether it is directed in row or column direction.
2. A symmetric adjacency set produces a symmetric co-occurrence matrix.

Those co-occurrence matrices are used to define *co-occurrence-based measures* to quantify information in an image I. Note that noise in an image is still considered to be information when using these measures. We provide here two of such measures:

$$M_{hom}(I) = \sum_{u,v \in \{0,1,\ldots,G_{\max}\}} \frac{\mathbf{C}_I(u, v)}{1 + |u - v|} \quad \text{(Homogeneity measure)} \tag{3.34}$$

$$M_{uni}(I) = \sum_{u,v \in \{0,1,\ldots,G_{\max}\}} \mathbf{C}_I(u, v)^2 \quad \text{(Uniformity measure)} \tag{3.35}$$

Informally speaking, a high homogeneity or uniformity indicates that the image I has more "untextured" areas.

Measures can also be defined by comparing the sum of all entries on or close to the main diagonal of the co-occurrence matrix to the sum of all entries in the remaining cells of the co-occurrence matrix.

Moment based Region Analysis

Assume a region $S \subset \Omega$ of pixel locations in image I

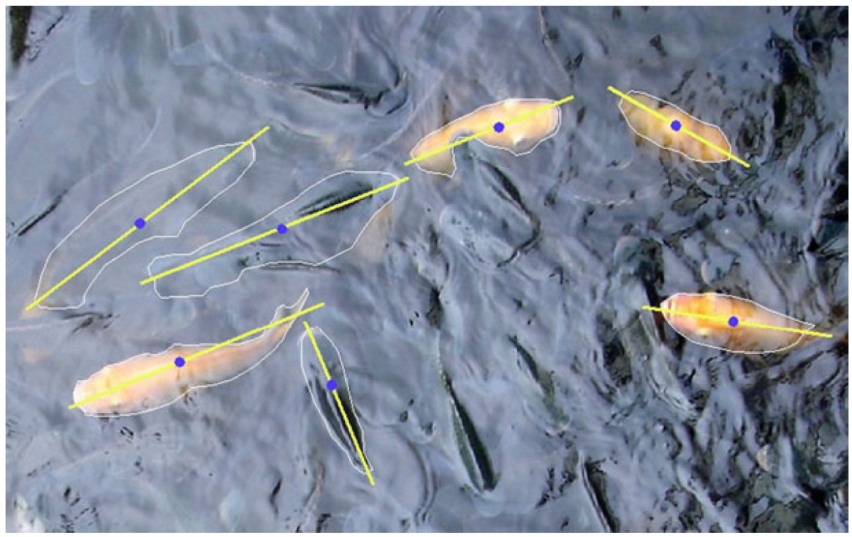


Fig. 3.30 Main axes and centroids for detected fish regions

For scalar image I. The moments of the region S in image I are defined by

$$m_{a,b}(S) = \sum_{(x,y)\in S} x^a y^b \cdot I(x, y) \tag{3.36}$$

for non-negative a, b. The sum of $a+b$ is order of the moment

There is only one moment

$$m_{0,0}(S) = \sum_{(x,y)\in S} I(x, y) \tag{3.37}$$

of order zero. If $I(x, y) = 1$ in S, then $m_{0,0}(S) = \mathcal{A}(S)$, the area of S. The moments of order 1,

$$m_{1,0}(S) = \sum_{(x,y)\in S} x \cdot I(x, y) \quad \text{and} \quad m_{0,1}(S) = \sum_{(x,y)\in S} y \cdot I(x, y) \tag{3.38}$$

define the *centroid* (x_S, y_S) of S as follows:

$$x_S = \frac{m_{1,0}(S)}{m_{0,0}(S)} \quad \text{and} \quad y_S = \frac{m_{0,1}(S)}{m_{0,0}(S)} \tag{3.39}$$

Note that the centroid depends on the values of I over S, not just on the shape of S.

The *central moments* of region S in image I are defined by

$$\mu_{a,b}(S) = \sum_{(x,y)\in S} (x - x_S)^a (y - y_S)^b \cdot I(x, y) \tag{3.40}$$

for non-negative integers a and b. The central moments provide a way to characterize regions S by features that are invariant with respect to any linear transform.

We only provide two examples here for such features. *Main axes* as shown in Fig. 3.30 are defined by an angle $\theta(S)$ (modulo π) with the positive x-axis and by being incident with the centroid. It holds that

$$\tan\big(2 \cdot \theta(S)\big) = \frac{2\mu_{1,1}(S)}{\mu_{2,0}(S) - \mu_{0,2}(S)} \tag{3.41}$$

Furthermore, the *eccentricity* $\varepsilon(S)$ of a region S is defined by

$$\varepsilon(S) = \frac{[\mu_{2,0}(S) - \mu_{0,2}(S)]^2 - 4\mu_{1,1}(S)^2}{[\mu_{2,0}(S) + \mu_{0,2}(S)]^2} \tag{3.42}$$

and characterizes the ratio of the main axis to the orthogonal axis of S. The eccentricity equals zero (in the ideal case) if S is a rotationally symmetric disk; then we have that $\mu_{2,0}(S) = \mu_{0,2}(S)$. A line segment has eccentricity one in the ideal case (and close to one when measured in an image). Rotational symmetric sets with the centroid at the origin also satisfy that $m_{1,1}(S) = 0$.

Example 3.7 (Accuracy of Recorded Marks) In computer vision we have cases that a special mark on a surface (e.g. a drawn cross or circle) needs to be accurately localized in a captured image I. There is a standard two-step procedure for doing so:

1. Detect the region S of pixels which is considered to be the image of the mark.
2. Calculate the centroid of the region S also using the values of I in S, defined by (3.39).

In this way, the position of the mark in I is determined with subpixel accuracy.

Example 3.8 (Achieving Rotation Invariance for Analysing a Region) When deriving features for an image region, it is often desirable to have *isotropic* features (i.e. invariant with respect to rotation). Moments support a four-step procedure for doing so:

1. Detect the region S of pixels which needs to be analysed.
2. Calculate the main axis, as defined by (3.41).
3. Rotate S so that its main axis coincides with the x-axis (or any other chosen fixed axis). In cases of $\mu_{1,1}(S) = 0$ or $\mu_{2,0} = \mu_{0,2}$ (i.e. rotation-symmetric shapes), no rotation is needed.
4. Calculate the isotropic features for the direction-normalized set as obtained after rotation.

Note that rotation in the grid causes minor deviations in values in a region S, and also its shape will vary a little due to limitations defined by the grid.

Detection of Lines and Circles

→ An edge detector is first used to map a given image into binary edge map. Then remaining all edge pixels are analysed to see if they are forming any line.

→ Read book for lines detection using Hough Transform

Circle Detection using Hough Transform

The idea of Hough transforms can be generalized: for geometric objects of interest, consider a parameterization that supports a bounded accumulator array. Insert pixels into this array and detect peaks. We illustrate this generalization for circles. For example, see Fig. 3.36 for an application where circle detection is very important. Stop signs are nearly circular.

In this case, a straightforward parameterization of circles is fine for having a bounded accumulator array. We describe a circle by a centre point (x_c, y_c) and radius r. This defines a 3D Hough space.

Consider a pixel location $p = (x, y)$ in an image I. For a start, it may be incident with a circle of radius $r = 0$, being the pixel itself. This defines the point $(x, y, 0)$ in the Hough space, the starting point of the surface of a straight circular cone. Now assume that the pixel location $p = (x, y)$ is incident with circles of radius $r_0 > 0$. The centre points of those circles define a circle of radius r_0 around $p = (x, y)$. See Fig. 3.37, left. This circle is incident with the surface of the straight circular cone defined by (x, y) in the Hough space. To be precise, it is the intersection of this surface with the plane $r = r_0$ in the xyr Hough space.

For the radius r of circular borders in an image, we have an estimate for a possible range $r_0 \leq r \leq r_1$. The size of the images always defines an upper bound. Thus, the 3D Hough space for detecting circles reduces to a "layer" of thickness $r_1 - r_0$ over Ω. Thus, we do have a bounded Hough space.

This Hough space is discretized into 3D cells, defining a 3D accumulator array. At the beginning of a detection process, all counters in the accumulator array are set back to zero. When inserting a pixel location (x, y) into the accumulator array, all counters of the 3D cells that are intersected by the surface of the cone of pixel location (x, y) increase by one.

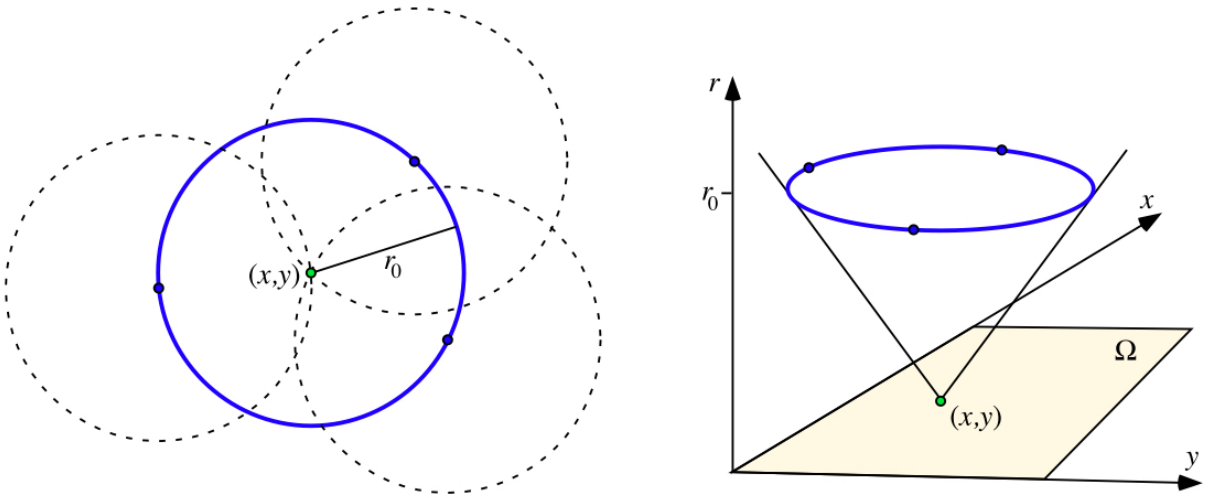


Fig. 3.37 *Left*: Pixel location (x, y) and location of all centre points of circles of radius $r_0 > 0$ around pixel (x, y). *Right*: A pixel location (x, y) defines the surface of a right circular cone in the Hough space

For example, when inserting three non-collinear pixel locations into the 3D Hough space, we generate three surfaces of right circular cones. We know that three non-collinear points determine uniquely a circle in the plane. Thus, all the three surfaces will intersect in one point only, defining the parameters of the uniquely specified circle.

As in case of line detection, now we need to detect peaks in the 3D accumulator array. Multiple circles in a given image usually do not create much interference in the 3D accumulator array, and there is also no "endpoint problem" as in case of line detection.