Statistical properties of digitized line segments

Vossepoel, A.M.; Smeulders, A.W.M.

Publication date
1981

Published in
Computer Graphics 81: proceedings of the international conference

UvA Keur
UB Groningen
Broerstraat 4
9700 AN Groningen

A078327075 ISN: 681393 RAPDOC (R) **OPGEHAALD** NCC/IBL

Verzoeke te behandelen voor: 27-04-2005 **Ingediend door:** 0004/9999
Datum en tijd van indienen: 13-04-2005 20:31 **Datum plaatsen:** 13-04-2005 20:31 **Type aanvrager:** UKB **I.D.:** UVA KEUR (UB GRONINGEN)

PPN: 193065665

Computer graphics 81 : proceedings of the international conference. (1981)Online Conferences Ltd. 1981 Northwood Online Publications

Gewenst: **Electronisch leveren (LH=N)**

Pagina's:
471-483

Opmerking:
arno ID: 135392

HAAFF	STARIN	12/151	beschikbaar

1. ○ origineel gestuurd
2. ○ fotokopie gestuurd
3. ○ overige
4. ○ nog niet aanwezig
5. ○ niet aanwezig
6. ○ niet beschikbaar
7. ○ uitgeleend
8. ○ wordt niet uitgeleend
9. ○ bibliografisch onjuist
10. ○ bij de binder

Fakturen zenden aan: Rijksuniversiteit Groningen
Bibliotheek, Uitleenbureau
Postbus 559
9700AN Groningen

STATISTICAL PROPERTIES OF DIGITISED LINE SEGMENTS

A M Vossepoel
Leiden University Medical Center, Netherlands

A W M Smeulders
Delft University of Technology, Netherlands

Digitising straight line segments of finite length on a grid results in a specific type of chain code strings. A set of linearity conditions must be fulfilled if a chain code string is to represent a straight line segment.

Conversely, a chain code string that fulfills these linearity conditions will always represent a set of straight line segments. Within limits, posed by the grid points, changes in position and orientation of the original line segment don't affect the generated chain code string. But within the set of line segments generating the same code string the length will vary. A statistical analysis per set yields an unbiased estimate of the original line segment's length.

By application of the linearity conditions to chain code strings of curves in various connectivities (assuming 4, 6 or 8 neighbours), the string is split up into substrings that each can represent a straight line segment. This means a polygonal approximation of the original curve. The length of the curve can then be evaluated with high accuracy using the unbiased length estimates of the individual straight line segments between vertices.

By applying the described method to digitised circular objects with random size and position its accuracy and reproducibility are demonstrated and compared with those of other methods.

INTRODUCTION

At present, various methods are known to estimate the length of curves and straight line segments digitized on a grid. Usually lines are represented by vector elements that connect the points of a rectangular, or sometimes hexagonal grid. In the rectangular grid each point may be defined to have 4 or 8 neighbours, resulting in 4 or 8 different vector elements. In the usual coding scheme, the vector element in the positive x-direction is coded 0, and the code is incremented with counterclockwise rotation [1].

One class of length estimation methods assigns a distinct length to each code. In the oldest method, for an 8-connected grid, the grid constant u is assigned to the even codes and $u.2^{1/2}$ to the odd ones. A better approach is to compute the probability and expected length of odd and even codes, randomly positioning the line segment [2]. Alternatively, a least-squares adaptation to lines of infinite length [3,4] results in coefficients different from the ones reported earlier [2]. The metrication error is further reduced by correcting for the number of unequal consecutive codes or corners [3].

Another class of methods is based upon polygonal approximations of digitised line segments. The vertices of the polygonal approximation are chosen arbitrarily with a fixed number of codes between them [3], or according to an irregularity criterion [5]. The length of the associated line segment is then computed as the Euclidean distance between the vertices.

In this paper, a synthesis between the two classes of methods is presented. The method mentioned first [2] is generalized to express the probability of a vector and the consecutive codes associated with it, even if more than one single code is involved. This approach results in an unbiased length estimate for the straight line segment associated with a given number of consecutive codes.

To an arbitrary code string one must apply linearity conditions [6,7] in order to find the vertices of a polygonal representation, analogous to a polygonal approximation [5]. Having found the vertices, the unbiased length estimates of the code strings between the vertices are used, instead of the Euclidean distances.

The linearity conditions can be summarized as follows:
- no more than 2 different codes are involved;
- the difference between these codes is 1 or connectivity-1;
- the minority code always appears isolated;
- no more than 2 runlengths of majority codes are involved;
- the difference between these runlengths is 1 or less;
- the minority runlengths of the majority codes appear isolated;
- etc., replacing "majority codes" by "majority runlengths".

DIGITISATION

From the linearity conditions follows that any straight line can only be digitised into two adjacent chain codes. Therefore it is sufficient to consider only the digitisation of lines with directions between those of the basic vectors coded 0 and 1. The direction of any other line can be brought into this range by rotation over $\pi/2$ (rectangular grid) or $\pi/3$ (hexagonal grid), or by reflection with respect to a basic vector direction. Likewise, a line can always be translated such that it passes through an entry window, the difference vector of the basic vectors coded 1 and 0, originating from an assumed origin. In fig. 1 its end points P and Q are indicated for various connectivities c: c=4, c=6 and c=8, respectively. The origin is assumed at the intersection of the lines WQ and VP, but has been omitted in the figure for the sake of clarity.

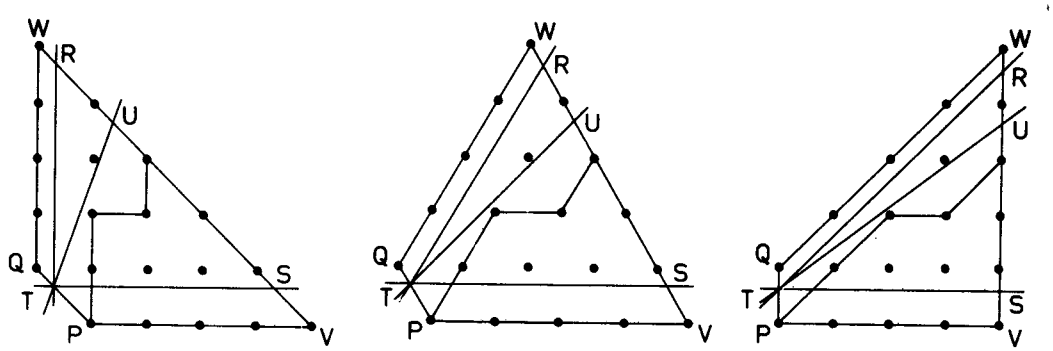

Figure 1: Length of a straight line segment, represented by 4 vector codes (1101, n=4) in 4- (a), 6- (b), and 8-connectivity (c).

We define a strip here as the part of the plane between adjacent grid lines running parallel to the entry window and through the grid points. A line intersecting the entry window at an angle τ with the direction of code 0, $0<\tau<2\pi/c$, will then generate one code for every intersection with a strip. In fig. 1 the situation in which the line TU intersects 4 strips between PQ and VW, thus generating 4 codes, is depicted. The line is digitised by connecting the highest grid points below the line on both sides of each strip, or, in practice, by a standard line-tracking algorithm starting at P.

There are many strings of n codes that can be generated by varying the direction of the line between $\tau=0$ and $\tau=2\pi/c$, and by varying the position of the entry point (T) on the entry window (PQ). Even if this position is kept fixed, the number of different strings will be of the order of $n(n+1)/2$.

In order to reduce this large number of possibilities, we consider code strings as equivalent if they have the same number of codes n, the same number of odd (i.e. with value 1) codes m, and the same

number of corners n_c. Because of the linearity conditions, n_c can have only 3 different values for distinct values of n and m. The codes that belong to the minority will be distributed evenly over the string, which is equivalent to a maximal number of corners. If both ends of the code string consist of majority codes, the number of corners is twice that of the minority codes, but each minority code occurring at either end of the code string reduces the value of n_c by one.

We here replace n_c by the parameter k, defined as the (integer) number of codes 1 at both ends of a code string: k = code(1)+code(n).

Conversely, n_c is then given by: $n_c(n,m,k) = (n-1)-|(n-1)-(2m-k)|$,

The change of sign in the argument of the abs-function occurs when m=n/2 and k=1. For even-valued n we then get:

$n_c(n,n/2,0) = n_c(n,n/2,2)\ (= n-2)$,

which implies that the use of k instead of n_c gives a slightly more accurate description.

For distinct values of n and m, the value of the first code of the string is determined by the existence of an intersection between the line segment TU and the horizontal grid line through Q in the first strip:

code(1) = [e+t].

with e = |PT|/|PQ| (relative entry height), t = |US|/|RS| (relative exit height), and the brackets [] denoting the entier function. Likewise, the value of the last code is determined by the intersection, or not, of the line segment TU with a horizontal grid line in the last strip, adjacent to VW:

code(n) = [e+nt] - [e+(n-1)t].

The entier function only changes value at an integer value of its argument. Therefore, equating the arguments of the entier functions mentioned with one of their possible integer values will provide the boundaries of the domains with constant n, m and n_c:

e+nt = m (with 0<m<n+1), e+t = 1, and e+(n-1)t = m (with 0<m<n).

Beside these intermediate boundaries, there are, of course, the ultimate boundaries represented by t=0, e=0, t=1 and e=1. In fig. 2, the quantity e+nt is plotted as a function of e for the various boundaries by which the domains of the segments with the same values of n, m, and n_c are delineated. The resulting code strings are indicated in each domain. From the figure, one may also see that there are at most 3 different values of n_c possible for each value of n and m (represented by the square regions in the figure for 0<m<n).

These values occur in 4 different sub-regions, separated by the second and third boundaries mentioned.

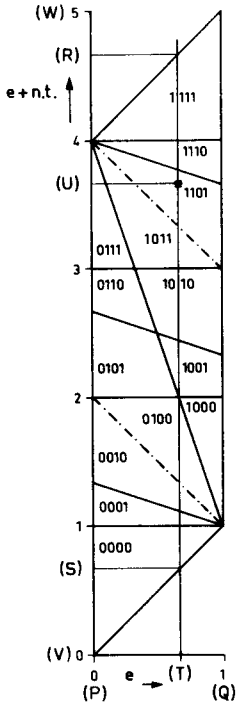

Figure 2: Code strings generated by the segment TU in figure 1 as a function of the position of T on the segment PQ and of U on the segment SR. The position of the point U in fig. 1 is indicated by an asterisk. The dotted lines separate regions with different code-strings that have the same number of odd codes, also at both ends.

LENGTH EVALUATION

If we want to know the segment length s, expressed as a multiple of the grid constant u, averaged over various orientations (τ) and entry heights (e), and its variance, we should be able to integrate three moment generating functions over these variables: the weight (h_0) (which, after normalization, is the probability density), the weighted length (h_1) and the weighted length squared (h_2). The resulting integrals over a domain with distinct values of n,m and k are called G_0, G_1 and G_2, respectively. Then the average length L(n,m,k) of all line segments that can be digitised into these distinct values is given by:

$$L(n,m,k) = G_1/G_0, \text{ with}$$

$$\text{standard deviation: } (G_2/G_0-(G_1/G_0)^2)^{1/2}.$$

In this scheme, normalization is performed by dividing G_1 and G_2 by G_0, the integrated weight function, so there is no need for normalizing the weight function itself.

Assuming evenly distributed position and orientation of straight lines on the grid, the variables τ and e show a uniform probability density. The number of codes n per (arbitrary) unit length v of the straight line depends on the angle φ between the line and a line perpendicular to the earlier defined strips:

$$nu/v = \cos(\varphi).$$

On the other hand, the length s of the line segment TU (cf. fig. 1), measured in units u, is given by:

$$s = nq(c)/\cos(\varphi),$$

in which q(c) denotes the strip width, which depends on the connectivity:

$$q(4)=2^{-1/2},\ q(6)=(1/2).3^{1/2}, \text{ and } q(8)=1.$$

Therefore, assuming n and c constant, the code generating probability

per unit length of a straight line segment is apparently inversely proportional to its length s, cf.[2].

In the evaluation of the finite integrals involved, it turns out to be convenient to express the angle τ as a function of the length t. This involves the differential substitution:

$d\tau = (d\tau/dt)dt$, with: $d\tau/dt = 1/(dt/d\tau)$ if $dt/d\tau = 0$.

If we define the angle σ between the direction of the strips and the direction of code 0, the length t is found by applying the sine rule in the triangle STU:

$t = \sin(\tau)/\sin(\tau+\sigma) \longrightarrow dt/d\tau = \sin(\sigma)/(\sin(\tau+\sigma))^2$;

Likewise, applying the sine rule in the triangle STU again:

$\sin(\tau+\sigma) = \sin(\sigma)/s$, substitution of which gives: $d\tau/dt = \sin(\sigma)/s^2$.

So, if we use s as a weight function, the following expression results for the functions $h_i(t)$ to be integrated over t:

$h_i(t) = \sin(\sigma)/(s(t))^{(3-i)}$,

with $s(t) = (t^2+1-2t.\cos(\sigma))^{1/2}$

In order to evaluate, e.g., the average length of all segments that result in code 0, $h_1(t)$ should first be integrated over $0<t<1-e$, and the result then over $0<e<1$. After application of the same procedure to $h_0(t)$ the average length is the quotient of the two results.

As a result of the first integration over t, the primitive functions of $h_i(t)$ are $H_i(t)$:

$H_0(t) = (t-\cos(\sigma))/(s(t)\sin(\sigma))$;

$H_1(t) = n.\arctan((t-\cos(\sigma))/(\sin(\sigma)))$;

$H_2(t) = (n^2)\sin(\sigma)\log(s(t)+t-\cos(\sigma))$.

After the first integration over t, we have to substitute the integration boundaries t(e) in the primitive functions $H_i(t)$ of $h_i(t)$ before integrating over e. For the intermediate boundaries between domains with distinct values of n, m and k it is more convenient to substitute these boundaries as e(t) in the integration variable e. In the second integration d(e(t)) may then be replaced by $(de/dt)_b dt$, $(de/dt)_b$ denoting the differential quotient along the boundary. This is a simple constant, because the equations describing the intermediate boundaries define a linear relationship between e and t. For the two ultimate boundaries of the first integration, where t=0 and t=1 respectively, substitution in $H_i(t)$ remains necessary because $(de/dt)_b$ is not defined.

The results of the (second) integration, that of $H_i(t)$ over e, are called $F_i(t)$:

$$F_0(t) = (de/dt)_b s(t)/\sin(\sigma);$$

$$F_1(t) = (de/dt)_b n((t-\cos(\sigma))\arctan((t-\cos(\sigma))/\sin(\sigma))-\sin(\sigma)\log(s(t)))$$

$$F_2(t) = (de/dt)_b (n^2)\sin(\sigma)((t-\cos(\sigma))\log(s(t)+(t-\cos(\sigma)))-s(t)).$$

Since one pair of integration boundaries has already been substituted into these primitive functions, as $(de/dt)_b$, the original integration boundaries are now reduced to four points in the e-t-plane. The value of the finite integral over the domain of all code strings with distinct values of n, m, and k is then found by substitution of the values of t at the four corners of the integration domain:

$$G_i(n,m,k) = F_i(t_1)-F_i(t_2)+F_i(t_3)-F_i(t_4)$$

in which t_1 denotes the value of t at the first quadrant corner of the integration domain, t_2 at the second quadrant corner, etc. The ultimate integration boundaries of t, t=0 and t=1, present an exception to this expression: in these cases one should replace $F_i(t_3)-F_i(t_4)$ by $H_i(0)$, and $F_i(t_1)-F_i(t_2)$ by $H_i(1)$, respectively.

For $0<m<n$ the following expressions for $G_i(n,m,k)$ result:

$$G_i(n,m,0) = 2F_i(m/(n-1))-F_i((m+1)/n)-F_i((m-1)/(n-2));$$

$$G_i(n,m,1) = 2(F_i(m/n)+F_i((m-1)/(n-2))-F_i(m/(n-1))-F_i((m-1)/(n-1)));$$

$$G_i(n,m,2) = 2F_i((m-1)/(n-1))-F_i((m-1)/n)-F_i((m-1)/(n-2)).$$

By means of these expressions we can compute an unbiased estimate of the average length L(n,m,k) of all straight line segments that may be represented by a string of codes, provided the string can indeed represent a straight line segment:

$$L(n,m,k) = G_1/G_0, \text{ with standard deviation: } (G_2/G_0-(G_1/G_0)^2)^{1/2},$$

in which the arguments of G_i are omitted for the sake of clarity.

The resulting values of L(n,m,k) are summarized in table 1 for a number of values of n. Besides, a graphic impression of the application of some of the formulas derived above is presented in figure 3. Essentially, this figure is a representation of figure 2, with n=4 also, in which the area of the domains is made proportional to the integrated weight function $G_0(n,m,k)$. This is accomplished by taking for the ordinate, instead of t:

$$(H_0(t)-H_0(0))/(H_0(1)-H_0(0)).$$

The values of t are still indicated on the vertical axis (e=0) on a

non-linear scale. On the line segment e=1, some values of s(t) are indicated, reflecting the decreasing probability of segments of increasing length.

TABLE 1.

Relative probability or weight w, length s (=L/n), and variation coefficient d for some numbers of codes n, as a function of n, the number of odd codes m, and the number k of odd codes at either end of the code string. The number of corners is denoted by n_c.

				connectivity = 4			connectivity = 6			connectivity = 8		
n	m	k	n_c	w	s	d	w	s	d	w	s	d
1	0	0	0	0.500	0.785	10.2%	0.500	0.907	4.3%	0.586	1.059[a]	7.0%
1	1	2	0	0.500	0.785	10.2%	0.500	0.907	4.3%	0.414	1.183[a]	10.0%
1	total			1.000	0.785	10.2%	1.000	0.907	4.3%	1.000	1.111	8.4%
n	m	k	n_c	w	s	d	w	s	d	w	s	d
2	0	0	0	0.207	0.837	10.0%	0.232	0.930	4.2%	0.334	1.019	2.2%
2	1	1	1	0.586	0.749	7.0%	0.536	0.887	2.9%	0.504	1.113	7.5%
2	2	2	0	0.207	0.837	10.0%	0.232	0.930	4.2%	0.162	1.292	5.9%
2	total			1.000	0.785	8.4%	1.000	0.907	3.5%	1.000	1.111	5.9%
n	m	k	n_c	w	s	d	w	s	d	w	s	d
3	0	0	0	0.118	0.885	7.6%	0.146	0.949	3.3%	0.229	1.009	1.0%
3	1	1	1	0.178	0.772	7.2%	0.173	0.897	3.0%	0.209	1.041	2.5%
3	1	0	2	0.204	0.739	6.4%	0.182	0.882	2.6%	0.189	1.075	4.4%
3	2	2	2	0.204	0.739	6.4%	0.182	0.882	2.6%	0.148	1.165	6.1%
3	2	1	1	0.178	0.772	7.2%	0.173	0.897	3.0%	0.127	1.228	4.8%
3	3	2	0	0.118	0.885	7.6%	0.146	0.949	3.3%	0.099	1.333	4.0%
3	total			1.000	0.785	7.0%	1.000	0.907	2.9%	1.000	1.111	3.9%
n	m	k	n_c	w	s	d	w	s	d	w	s	d
4	0	0	0	0.081	0.914	5.9%	0.106	0.961	2.6%	0.174	1.005	0.6%
4	1	1	1	0.074	0.823	6.1%	0.080	0.919	2.6%	0.111	1.021	1.2%
4	1	0	2	0.178	0.772	7.2%	0.173	0.897	3.0%	0.209	1.041	2.5%
4	2	2	2	0.052	0.736	2.5%	0.046	0.878	0.9%	0.049	1.063	1.7%
4	2	1	3	0.229	0.713	1.0%	0.191	0.869	0.4%	0.169	1.117	2.7%
4	2	0	2	0.052	0.736	2.5%	0.046	0.878	0.9%	0.035	1.186	2.4%
4	3	2	2	0.178	0.772	7.2%	0.173	0.897	3.0%	0.127	1.228	4.8%
4	3	1	1	0.074	0.823	6.1%	0.080	0.919	2.6%	0.056	1.281	3.4%
4	4	2	0	0.081	0.914	5.9%	0.106	0.961	2.6%	0.071	1.354	3.0%
4	total			1.000	0.785	5.5%	1.000	0.907	2.4%	1.000	1.111	2.7%

[a] Cf. [1]

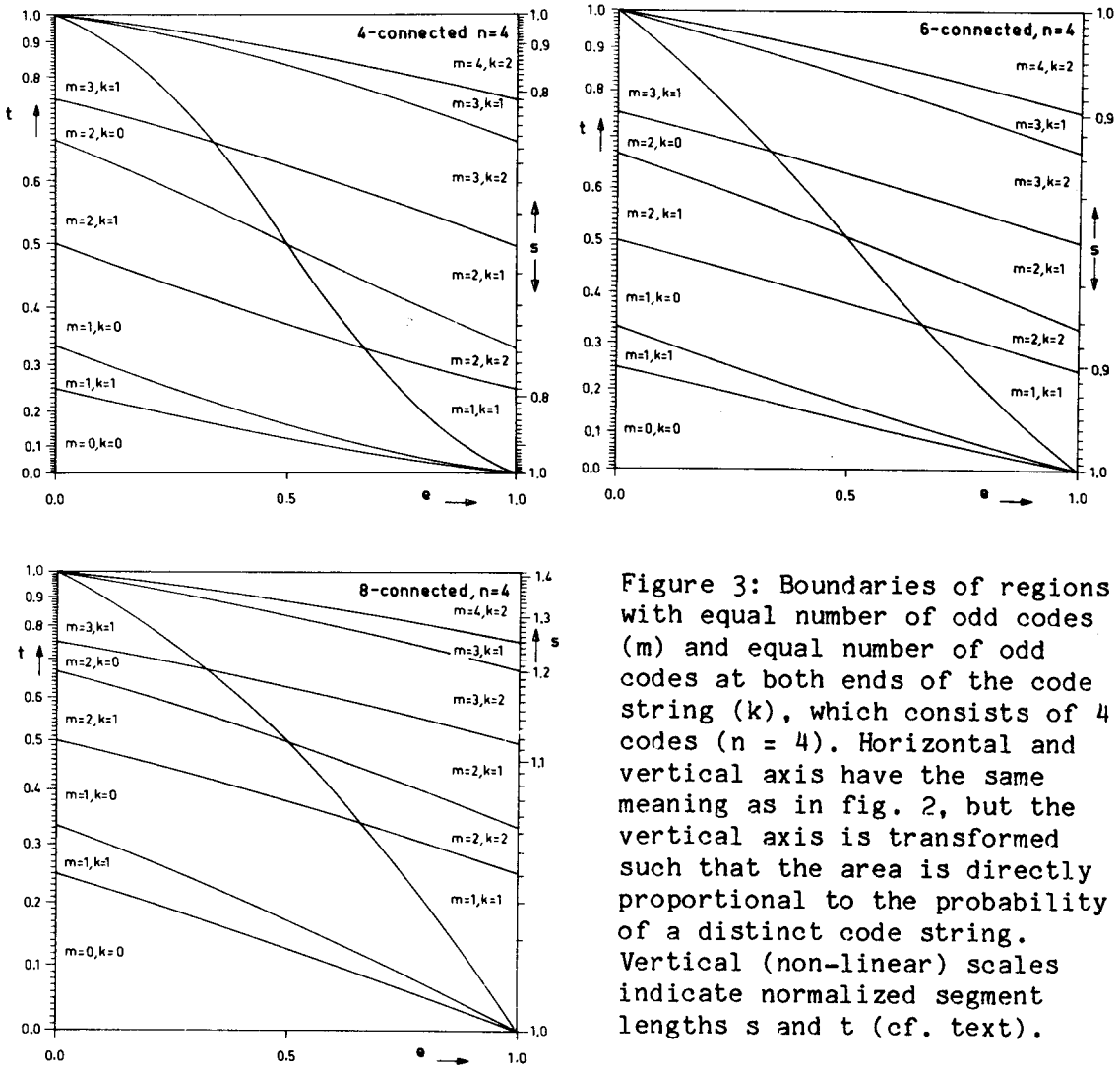

Figure 3: Boundaries of regions with equal number of odd codes (m) and equal number of odd codes at both ends of the code string (k), which consists of 4 codes (n = 4). Horizontal and vertical axis have the same meaning as in fig. 2, but the vertical axis is transformed such that the area is directly proportional to the probability of a distinct code string. Vertical (non-linear) scales indicate normalized segment lengths s and t (cf. text).

COMPARISON WITH OTHER METHODS

If we want to compute the segment length K by means of a relation linear in n, m and n_c, we may adopt a least-squares approximation to evaluate the coefficients a_n, b_n and c_n for a distinct value of n in:

$$K(n,m,n_c) = na_n + mb_n + n_c c_n,$$

in which a_n represents the length assigned to all codes, b_n the length correction for odd codes, and c_n the correction for the corner count n_c.

For the normal equations used in this least-squares approach we have adopted the same dimensionality as used in [3]:

$$\sum (nG_0(K/L-1)) = 0,$$

$$\sum (mG_0(K/L-1)) = 0,$$

$$\sum (n_c G_0(K/L-1)) = 0,$$

in which the summations are performed over the different values of m and n_c, given n, and the arguments of the functions G_0, K and L have been omitted for the sake of clarity. Each term of the equations might as well have been multiplied or divided by L, in order to minimize the absolute or the relative variance, respectively.

In 4- or 6-connectivity, any difference in length assigned to odd or even codes is artificial, so we may remove the second equation from the set, and put $b_n=0$, for $n=\infty$ and c=4 giving the same result as in [3], An analogous situation occurs in 8-connectivity when the equations account for the number of corners, but the length assigned to the odd-coded segments is a predefined multiple q ($q_8=2^{1/2}$) of the length assigned to the even-coded segments. Then $b_n=a_n(q-1)$, and the second equation should also be removed from the set.

In 8-connectivity, a special case of these equations occurs when the corners are not taken into account. Then one should restrict the set of equations to the first two and put $c_n=0$, for n=1 giving the same results as [2].

TABLE 2.

Coefficients a_n (average length per code), b_n (odd code correction), and c_n (corner correction) used in the computation of the segment length

$K=a_n.n+b_n.m+c_n.n_c$ for n=1000;

d_n denotes the resulting average variation coefficient of K, d_n' the same for the unbiased length estimate L, and e_n' the average absolute error in L.

connectivity	constraints	a_n	b_n	c_n	d_n	d_n'	e_n'
4	$b_n=0$	0.948[a]	0.000	-0.278[a]	2.3 %[a]	0.021 %	0.184
6	$b_n=0$	0.977	0.000	-0.131	1.0 %	0.010 %	0.096
8	$b_n=a_n.(2^{1/2}-1)$	0.984	0.408	-0.085	0.8 %	0.009 %	0.114
8	$c_n=0$	0.948	0.392	0.000	2.3 %	0.009 %	0.114
8		0.980	0.426	-0.091	0.7 %	0.009 %	0.114

[a] Cf. [3]

In table 2 the values of a_n, b_n and c_n are presented for n=1000. This number of codes is large enough to make the results comparable with those described (for 4-connectivity) for infinite n [3]. The results

for 6- and 8-connectivity are also presented, using the constraints mentioned before, or not. One may see from the table that the constraint $c_n=0$ in 8-connnectivity results in the same values for the average length per code a_n and the variation coefficient d_n as in 4-connectivity.

Note that the absolute lengths in 6-connectivity have to be multiplied by $(4/3)^{1/4}$, in order to compensate for the higher density of grid points, and consequently shorter grid constant u in these grids.

By applying the linearity conditions to an increasing number of codes until failure one may construct a polygonal representation of a code string of arbitrary shape and length, assuming a vertex occurring just after the last code. The next code in turn may be taken as the first code of the next string. The process may be repeated until the end of the string is reached. For closed contours that contain a "sharp" corner, i.e., two adjacent codes representing non-adjacent directions, it makes sense to start at this "sharp" corner.

In order to test this polygon approach we generated circular objects on rectangular and hexagonal grids. The radius of the circles was given 32 uniformly distributed random values between 1.5u and 17.5u. For each value of the radius the centroid position was uniformly randomized over 64 positions. This was done by adding p times basic vector coded 0 and q times basic vector coded 1, p and q being uniformly distributed variables with values between 0 and 1.

Each circular object generated was digitised into an inner and into an outer contour chain code string by means of a simple contour tracking algorithm. The resulting chain code strings were both segmented into polygons, each string once forward and once backward. Then we applied 3 methods of length estimation to each string: the naive method [1], the corner count method [3] and the proposed method. In the naive method, we assigned a length of 1 to each code, except the odd ones with c=8, to which we assigned $2^{1/2}$. In the corner count method, we used the one with no constraints for c=8, cf. the lowest line of table 2.

The perimeter length was computed as the average of the 4 contour lengths (inner, outer, forward and backward). The difference between the real perimeter length of the object and the one estimated by each method was averaged over the 64 centroid positions, resulting in the average systematic error for a distinct radius. In the naive method, the length estimate is biased, so the systematic error had to be expressed as a percentage of the perimeter length. The other methods showed little correlation between systematic error and perimeter length, so there we could adopt the average over all 32 radius values for the systematic error. For the root-mean-square error in the average perimeter values the correlation with perimeter length was also slight. Therefore, this quantity could also be averaged over all radius values. The results of the test are presented in table 3.

TABLE 3

Average errors in the perimeter lengths of circular objects on grids with 4-, 6- and 8-connectivity.

Radius and centroid position of the objects have been uniformly randomised within distinct ranges (see text).

	average systematic error			average r.m.s. error		
connectivity:	4	6	8	4	6	8
Naive method [1]:	27%	10%	5%	1.1	0.7	0.5
Corner count method [3]:	1.3	0.7	0.4	0.7	0.4	0.4
Proposed method:	-0.2	-0.1	-0.2	0.4	0.3	0.3

DISCUSSION

The length estimate derived in [2] for one single code will remain unbiased as long as the curve represented by a code string is isotropic. It is evident that with increasing n the variation coefficient (d) will remain at the value for n=1, whereas in the proposed method this coefficient decreases towards 0.

From table 1 appears that the length estimation of coded linear segments is most accurate with c=6 for short segments. Looking at the variation coefficients d_n' in table 2, the accuracy of long linear segments seems optimal with c=8. However, for large n the average absolute error e_n' in the segment length is decreasing towards a constant with increasing number of codes. This constant value is smaller for c=6 than for c=8, even after compensation for the higher density of grid points, and consequently smaller grid constant u in 6-connectivity.

From tables 1 and 2 we arrive at the conclusion, other than [3], that with each method mentioned 8-connected grids are better suited for length measurements than 4-connected grids. In this comparison we have accounted for the fact that the number of codes representing a segment with distinct length is larger by a factor $2^{1/2}$ for c=4 than it is for c=8.

As for the various least-squares approaches, one may see from table 2 that with c=8 application over all three variables leads to smaller variation coefficients than application with constraints.

For large numbers of codes, the greatest difference between the least squares approach and the presented length estimation method is also in the resulting relative error. In the first approach the error remains finite, even if the number of codes is infinite, whereas in

our approach the absolute error decreases towards a constant for large n, leading to a vanishing relative error for an infinite number of codes. In this case also, the difference is caused by the forced linear relationship of the least squares approach.

The test with the randomly generated circular objects shows that the proposed method is more accurate for computing the length of curved contours as well. The difference with the naive method is really striking in this respect. The reproducibility of the proposed method is also better than that of the other methods. The accuracy and reproducibility of the contour length estimation in 6-connectivity are slightly better than in 8-connectivity with the proposed method, as opposed to the other methods. These results are confirmed by other experiments [8].

A by-product of the test were the differences in length between the inner and outer contour chain code string. In 4-, 6- and 8-connectivity, they turned out to be 7.7, 6.3 and 5.6 respectively, virtually irrespective of the length estimation method used. The polygonal representation also has potentialities for increasing the coding efficiency, e.g. by assigning to each segment between two vertices one single code.

REFERENCES:

[1] H. Freeman: Boundary Encoding and Processing. In: Picture Processing and Psychopictorics, B.S. Lipkin, A. Rosenfeld (eds.), (New York, Academic Press, 1970) pp.241-266.

[2] F.C.A. Groen and P.W. Verbeek: Freeman-Code Probabilities of Object Boundary Quantized Contours. Comput. Graph. Image Process. 7 (1978) 391-402.

[3] D. Proffitt and D. Rosen: Metrication Errors and Coding Efficiency of Chain-Encoding Schemes for the Representation of Lines and Edges. Comput. Graph. Image Process. 10 (1979) 318-332.

[4] T.J. Ellis, D.Proffitt, D. Rosen and W. Rutkowski: Measurement of the Lengths of Digitized Curved Lines. Comput. Graph. Image Process. 10 (1979) 333-347.

[5] Z. Kulpa: Area and Perimeter Measurement of Blobs in Discrete Binary Pictures. Comput. Graph. Image Process. 6 (1977) 434-451.

[6] R. Brons: Linguistic Methods for the Description of a Straight Line on a Grid. Comput. Graph. Image Process. 3 (1974) 48-62.

[7] L.D. Wu: On the Freeman's Conjecture about the Chain Code of a Line. Proc. 5th Int. Conf. on Pattern Recognition, Miami Beach, Fla., Dec.1-4, 1980. (IEEE, New York, 1980) pp.32-34.

[8] A.W.M. Smeulders, A.M. Vossepoel, J. Vrolijk, J.S. Ploem and C.J. Cornelisse. Some Shape Parameters for Cell Recognition. In: Pattern Recognition in Practice. E.S. Gelsema and L.N. Kanal (eds.), (North-Holland, Amsterdam, 1980) pp.131-142.