function c = fchcode(b, conn, dir)

%FCHCODE Computes the Freeman chain code of a boundary.

% C = FCHCODE(B) computes the 8-connected Freeman chain code of a

% set of 2-D coordinate pairs contained in B, an np-by-2 array. C

% is a structure with the following fields:

%

% c.fcc = Freeman chain code (1-by-np)

% c.diff = First difference of code c.fcc (1-by-np)

% c.mm = Integer of minimum magnitude from c.fcc (1-by-np)

% c.diffmm = First difference of code c.mm (1-by-np)

% c.x0y0 = Coordinates where the code starts (1-by-2)

%

% C = FCHCODE(B, CONN) produces the same outputs as above, but

% with the code connectivity specified in CONN. CONN can be 8 for

% an 8-connected chain code, or CONN can be 4 for a 4-connected

% chain code. Specifying CONN=4 is valid only if the input

% sequence, B, contains transitions with values 0, 2, 4, and 6,

% exclusively.

%

% C = FHCODE(B, CONN, DIR) produces the same outputs as above, but,

% in addition, the desired code direction is specified. Values for

% DIR can be:

%

% 'same' Same as the order of the sequence of points in b.

% This is the default.

%

% 'reverse' Outputs the code in the direction opposite to the

% direction of the points in B. The starting point

% for each DIR is the same.

%

% The elements of B are assumed to correspond to a 1-pixel-thick,

% fully-connected, closed boundary. B cannot contain duplicate

% coordinate pairs, except in the first and last positions, which

% is a common feature of boundary tracing programs.

%

% FREEMAN CHAIN CODE REPRESENTATION

% The table on the left shows the 8-connected Freeman chain codes

% corresponding to allowed deltax, deltay pairs. An 8-chain is

% converted to a 4-chain if (1) if conn = 4; and (2) only

% transitions 0, 2, 4, and 6 occur in the 8-code. Note that

% dividing 0, 2, 4, and 6 by 2 produce the 4-code.

%

% ----------------------- ----------------

% deltax | deltay | 8-code corresp 4-code

% ----------------------- ----------------

% 0 1 0 0

% -1 1 1

% -1 0 2 1

% -1 -1 3

% 0 -1 4 2

% 1 -1 5

% 1 0 6 3

% 1 1 7

% ----------------------- ----------------

%

% The formula z = 4\*(deltax + 2) + (deltay + 2) gives the following

% sequence corresponding to rows 1-8 in the preceding table: z =

% 11,7,6,5,9,13,14,15. These values can be used as indices into the

% table, improving the speed of computing the chain code. The

% preceding formula is not unique, but it is based on the smallest

% integers (4 and 2) that are powers of 2.

% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins

% Digital Image Processing Using MATLAB, Prentice-Hall, 2004

% $Revision: 1.6 $ $Date: 2003/11/21 14:34:49 $

% Preliminaries.

if nargin == 1

dir = 'same';

conn = 8;

elseif nargin == 2

dir = 'same';

elseif nargin == 3

% Nothing to do here.

else

error('Incorrect number of inputs.')

end

[np, nc] = size(b);

if np < nc

error('B must be of size np-by-2.');

end

% Some boundary tracing programs, such as boundaries.m, output a

% sequence in which the coordinates of the first and last points are

% the same. If this is the case, eliminate the last point.

if isequal(b(1, :), b(np, :))

np = np - 1;

b = b(1:np, :);

end

% Build the code table using the single indices from the formula

% for z given above:

C(11)=0; C(7)=1; C(6)=2; C(5)=3; C(9)=4;

C(13)=5; C(14)=6; C(15)=7;

% End of Preliminaries.

% Begin processing.

x0 = b(1, 1);

y0 = b(1, 2);

c.x0y0 = [x0, y0];

% Make sure the coordinates are organized sequentially:

% Get the deltax and deltay between successive points in b. The

% last row of a is the first row of b.

a = circshift(b, [-1, 0]);

% DEL = a - b is an nr-by-2 matrix in which the rows contain the

% deltax and deltay between successive points in b. The two

% components in the kth row of matrix DEL are deltax and deltay

% between point (xk, yk) and (xk+1, yk+1). The last row of DEL

% contains the deltax and deltay between (xnr, ynr) and (x1, y1),

% (i.e., between the last and first points in b).

DEL = a - b;

% If the abs value of either (or both) components of a pair

% (deltax, deltay) is greater than 1, then by definition the curve

% is broken (or the points are out of order), and the program

% terminates.

if any(abs(DEL(:, 1)) > 1) | any(abs(DEL(:, 2)) > 1);

error('The input curve is broken or points are out of order.')

end

% Create a single index vector using the formula described above.

z = 4\*(DEL(:, 1) + 2) + (DEL(:, 2) + 2);

% Use the index to map into the table. The following are

% the Freeman 8-chain codes, organized in a 1-by-np array.

fcc = C(z);

% Check if direction of code sequence needs to be reversed.

if strcmp(dir, 'reverse')

fcc = coderev(fcc); % See below for function coderev.

end

% If 4-connectivity is specified, check that all components

% of fcc are 0, 2, 4, or 6.

if conn == 4

val = find(fcc == 1 | fcc == 3 | fcc == 5 | fcc ==7 );

if isempty(val)

fcc = fcc./2;

else

warning('The specified 4-connected code cannot be satisfied.')

end

end

% Freeman chain code for structure output.

c.fcc = fcc;

% Obtain the first difference of fcc.

c.diff = codediff(fcc,conn); % See below for function codediff.

% Obtain code of the integer of minimum magnitude.

c.mm = minmag(fcc); % See below for function minmag.

% Obtain the first difference of fcc

c.diffmm = codediff(c.mm, conn);

%-------------------------------------------------------------------%

function cr = coderev(fcc)

% Traverses the sequence of 8-connected Freeman chain code fcc in

% the opposite direction, changing the values of each code

% segment. The starting point is not changed. fcc is a 1-by-np

% array.

% Flip the array left to right. This redefines the starting point

% as the last point and reverses the order of "travel" through the

% code.

cr = fliplr(fcc);

% Next, obtain the new code values by traversing the code in the

% opposite direction. (0 becomes 4, 1 becomes 5, ... , 5 becomes 1,

% 6 becomes 2, and 7 becomes 3).

ind1 = find(0 <= cr & cr <= 3);

ind2 = find(4 <= cr & cr <= 7);

cr(ind1) = cr(ind1) + 4;

cr(ind2) = cr(ind2) - 4;

%-------------------------------------------------------------------%

function z = minmag(c)

%MINMAG Finds the integer of minimum magnitude in a chain code.

% Z = MINMAG(C) finds the integer of minimum magnitude in a given

% 4- or 8-connected Freeman chain code, C. The code is assumed to

% be a 1-by-np array.

% The integer of minimum magnitude starts with min(c), but there

% may be more than one such value. Find them all,

I = find(c == min(c));

% and shift each one left so that it starts with min(c).

J = 0;

A = zeros(length(I), length(c));

for k = I;

J = J + 1;

A(J, :) = circshift(c,[0 -(k-1)]);

end

% Matrix A contains all the possible candidates for the integer of

% minimum magnitude. Starting with the 2nd column, succesively find

% the minima in each column of A. The number of candidates decreases

% as the seach moves to the right on A. This is reflected in the

% elements of J. When length(J)=1, one candidate remains. This is

% the integer of minimum magnitude.

[M, N] = size(A);

J = (1:M)';

for k = 2:N

D(1:M, 1) = Inf;

D(J, 1) = A(J, k);

amin = min(A(J, k));

J = find(D(:, 1) == amin);

if length(J)==1

z = A(J, :);

return

end

end

%-------------------------------------------------------------------%

function d = codediff(fcc, conn)

%CODEDIFF Computes the first difference of a chain code.

% D = CODEDIFF(FCC) computes the first difference of code, FCC. The

% code FCC is treated as a circular sequence, so the last element

% of D is the difference between the last and first elements of

% FCC. The input code is a 1-by-np vector.

%

% The first difference is found by counting the number of direction

% changes (in a counter-clockwise direction) that separate two

% adjacent elements of the code.

sr = circshift(fcc, [0, -1]); % Shift input left by 1 location.

delta = sr - fcc;

d = delta;

I = find(delta < 0);

type = conn;

switch type

case 4 % Code is 4-connected

d(I) = d(I) + 4;

case 8 % Code is 8-connected

d(I) = d(I) + 8;

end