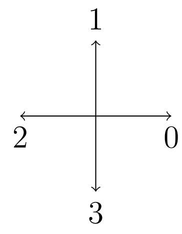
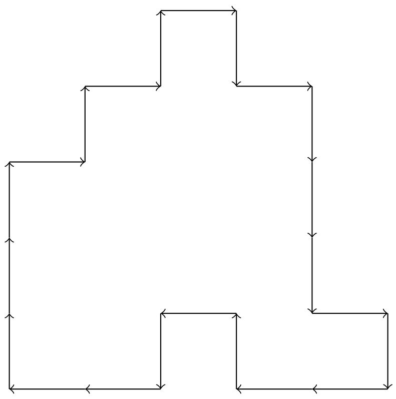
Tutorial: Freeman Chain Code

This tutorial is focused on shape representation by the Freeman chain code. This code is an ordered sequence of connected segments (of specific sizes and directions) representing the contour of the shape to be analyzed. The direction of each segment is encoded by a number depending on the selected connectivity (Fig. 1). In this example, the contour of the shape and its Freeman code are given in 4-connectivity.

Figure 1: Freeman chain code examples.

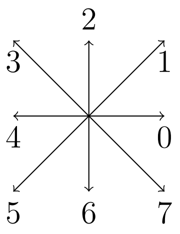


(a) 4-connected neighborhood.



(b) Example Freeman code in a 4-neighborhood.

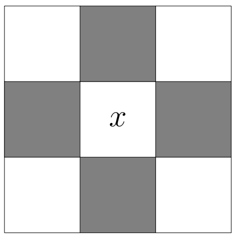
11101010303330322212322



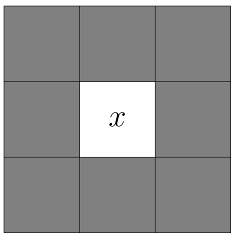
(c) 8-connected neighborhood.

Notations:

- y is 4-adjacent to x if $|y_1 - x_1| + |y_2 - x_2| \leq 1$.
- y is 8-adjacent to x if $\max(|y_1 - x_1|, |y_2 - x_2|) \leq 1$.
- $N_4(x) = \{y : y \text{ is 4-adjacent to } x\}$; $N_4^*(x) = N_4(x) \backslash \{x\}$.
- $N_8(x) = \{y : y \text{ is 8-adjacent to } x\}$; $N_8^*(x) = N_8(x) \backslash \{x\}$.



(a) $V_4(x)$



(b) $V_8(x)$

1 Shape contours

Before determining the Freeman chain code of the shape, it is necessary to extract its contour.

?

1. Generate or load a simple shape as a binary image A.
2. Extract its contour $C_4(A)$ ou $C_8(A)$ according to the 4-connectivity or the 8-connectivity, respectively:

$$\begin{aligned} x \in C_4(A) &\Leftrightarrow \exists y \in N_8(x) \quad y \in {}^cA \\ x \in C_8(A) &\Leftrightarrow \exists y \in N_4(x) \quad y \in {}^cA \end{aligned}$$

Informations

You can use the function bwperim with the appropriate connectivity number.

Informations

You can erode the object and subtract this erosion to it, with a structuring element that corresponds to N_4 or N_8 if you want to have 8 or 4 connectivity, respectively.

2 Freeman chain code

From the shape contours, the Freeman chain code can be calculated.

?

1. From the binary array of pixels, extract the first point belonging to the shape (from left to right, top to bottom).

   Informations

   You can use np.argwhere to locate the first point.

   Informations

   You can use find to locate the first point.

2. From this initial point, determine the Freeman chain code c (counterclockwise direction) using the N_4 or N_8 connectivity.

3 Normalization

The Freeman chain code is depending on the initial point and is not invariant to shape rotation. It is then necessary to normalize this code.

1. The first step consists in defining a differential code d from the code c :

   $$d_k = c_k - c_{k-1} (\text{mod } 4 \text{ or } 8)$$

   In this example, $d = 3003131331300133031130$;

2. The second step consists in normalizing the code d. We have to extract the lowest number p from all the cyclic translations of d.
   In this example, $p = 0013303113030031313313$.
   This Freeman chain code p is then independant from the initial point and invariant by rotations of angles $k * \pi/2$ radians ($k \in \mathbb{Z}$).

- Code above steps 1 and 2. Prototypes of functions are given.
- Validate your code by rotating the shape of $3\pi/2$ radians.

```python
def codediff(fcc, connectivity=8):
# computes differential code
```

Use numpy.roll for circularly shifting the freeman code array. Code a small function that tests all elements of array, one by one, in order to get the minimum of two arrays.

```matlab
function d=codediff(fcc,conn)
```

Use circshift for circularly shifting the freeman code array. Use polyval for transforming the array of values into a number, although rounding errors might occur, due to numerical approximation of floating numbers.

4 Geometrical characterization

4.1 Perimeter

From the Freeman chain code, it is possible to make different geometrical measurement of the shape.

Calculate the perimeter of the shape (take care of the diagonals).

4.2 Area

The area is evaluated by the following algorithm:

- The area is initialized to 0 and the parameter B is initialized to 0.
- For each iteration on the Freeman chain code c, the area and the parameter value B are incremented with the following rules:

8-code	area	B
0	-B	0
1	-B-0.5	1
2	0	1
3	B+0.5	1
4	B	0
5	B-0.5	-1
6	0	-1
7	-B+0.5	-1

Calculate the area of the shape in 8-connectivity. Note that the area does not correspond to the total number of pixels belonging to the shape.