M30242 Graphics and Computer Vision

Lecture 8 Shape Description

Introduction

- Image processing techniques make it possible to get some basic features from images, e.g., edges, lines, circles, etc.

- But we do not have methods to tell if two objects have different shapes, similar shapes or the same shape.
- The solutions to the above problems demand for the answers to the essential questions:
  – How can we represent and describe the shapes?
  – How do we calculate the similarity of shapes?

Cont'd

- Answers to these questions are still open research problems.
- In this lecture, we first look at the general problem of shape representation and description,
- then introduce two methods for shape description:
  - Freeman chain code, and
  - Fourier descriptors.

Shape Description

- Shape description is about choosing the “correct ways” to define shapes.
- In doing so, we wish the methods for shape description have the following (ideal) properties:
  - easiness – conceptually simple and easy to implement;
  - flexibility – applicable to as many shapes as possible;
  - discriminability – different shape will have different descriptions;
  - stability – invariant to geometric transformations (i.e., translation, rotation, and scale), and
  - repeatability – same description for the same object can be extracted every time.

Shape Descriptors

Shape description involves two tasks:

- Choosing a suitable feature (or features) as the representation of shapes – the problem of shape representation. For example,
  - using the external characteristics of a shape, such as its boundary, or,
  - using its internal characteristics, e.g., all the pixels comprising the shape.
- Choosing suitable parameters for the chosen features – the problem of shape description. E.g.,
  - boundary-based: length? number of concavities (and curvatures of the concavities)? …
  - area-based: area? centroid? moments about the axis of symmetry? …
- The parameters that describe a shape are called shape descriptors.

Shape Description is Hard

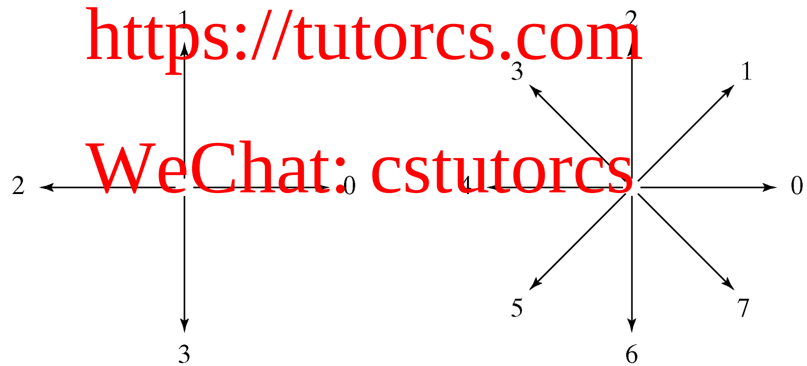
- Shape description is a very difficult problem, particularly in 3D.
- In remaining part of this lecture, we assume:
  - shapes are 2D,
  - boundaries are used for sgape description, and
  - boundaries can be extracted by, e.g., the Canny edge detector.

Boundary Description

- The simplest description of a boundary is using an ordered *list of edge points,* or similar (e.g., R-tables).
  - Not a compact representation.
  - Not an effective representation for subsequent image analysis.
- A better representation might be to fit an analytical curve (i.e., line segments, circular arcs, cubic splines) to the boundary points.
  - More compact and efficient representation for subsequent image analysis
  - Increases accuracy – errors in edge location are reduced through averaging (e.g., fitting a line to a set of edge points that lie along a line)
  - Not easy and repeatability might be very poor.

Freeman Chain Code

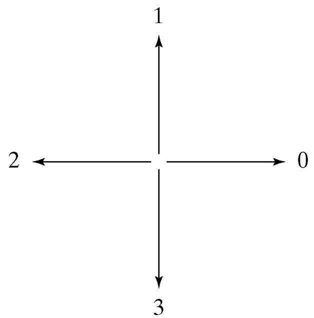
- A chain code describes a boundary by a connected sequence of straight-line segments of *specified step length and direction.*
- We may use 4 or 8 pre-defined directions:

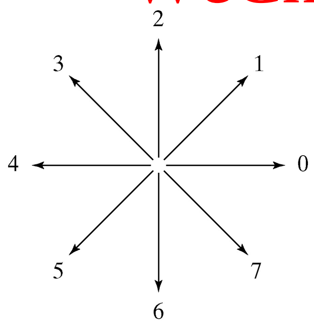


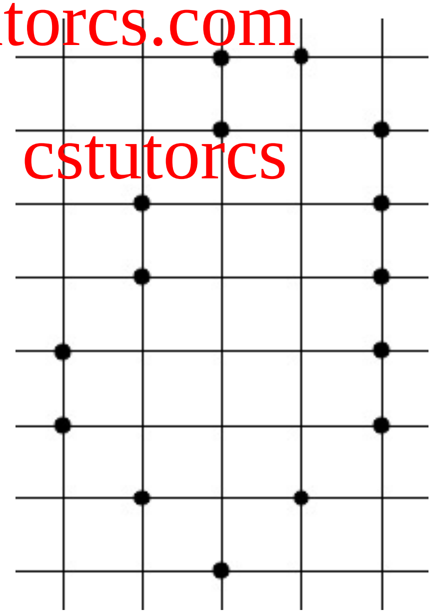
- With the directions and step length being defined, a shape can be described by “walking” around the boundary and taking note of the directions of the steps that one has to take.

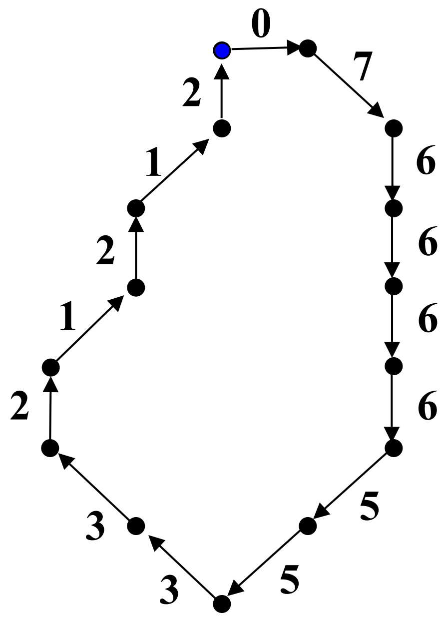
Freeman Chain Code

- Choose a *start point* on the boundary (e.g., the top-left corner) and go *clockwise* around the boundary, and
- Record the direction to the next edge point using one of the 4 or 8 discrete directions until the start point is reached.
- The result of this process is a string of numbers – the *Freeman chain code.*





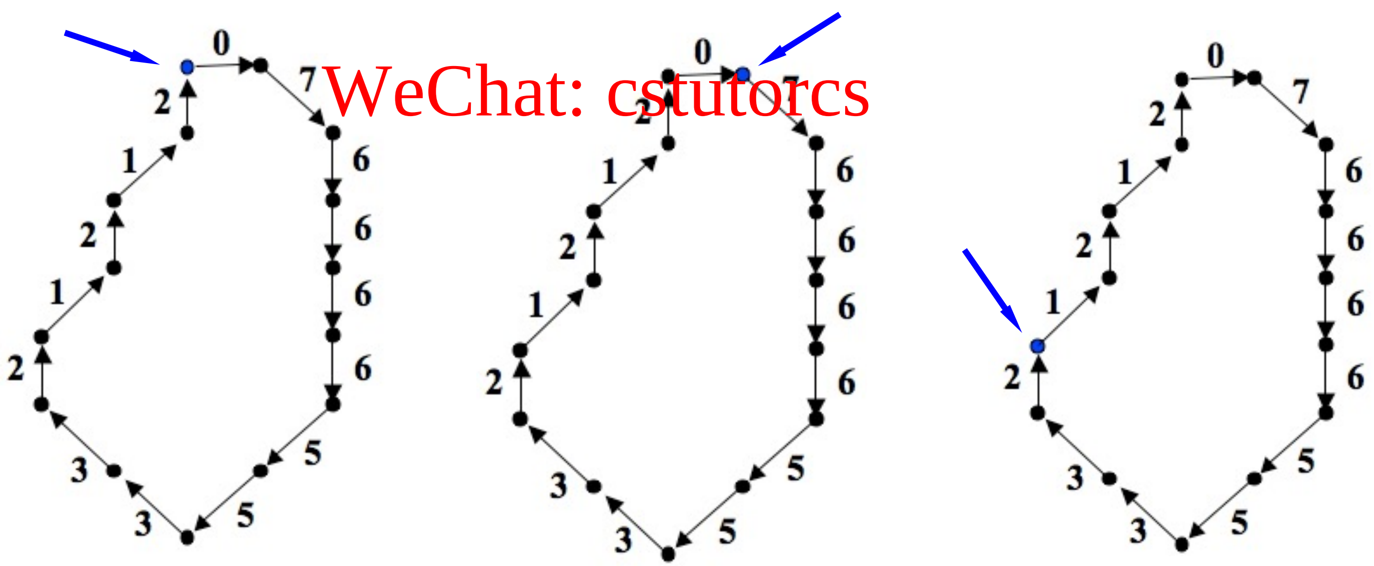




Chain code: 076666553321212

Limitations

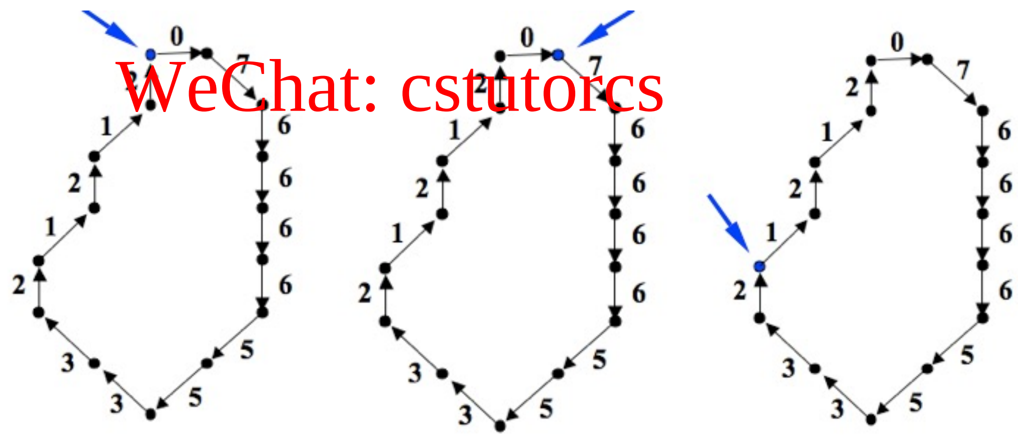
- The limitations of raw Freeman chain code is obvious: it *depends* on:
  - where we choose as the *start point,* and
  - *orientation* of the object.
- Such chain codes are said to be *variant to start point and orientation,* which are not ideal properties for shape description.



Chain code: 076666553321212 766665533212120 121207666655332

Chain Code Normalisation (1)

- The process that makes the chain code invariant to start point and orientation (to certain extent) is called *chain code normalisation.*
- The “raw” chain-code can be made invariant to starting point:
  - treat the code as a circular sequence, and
  - choose the *integer of minimum value* of all possible chain codes formed from the circular sequence.

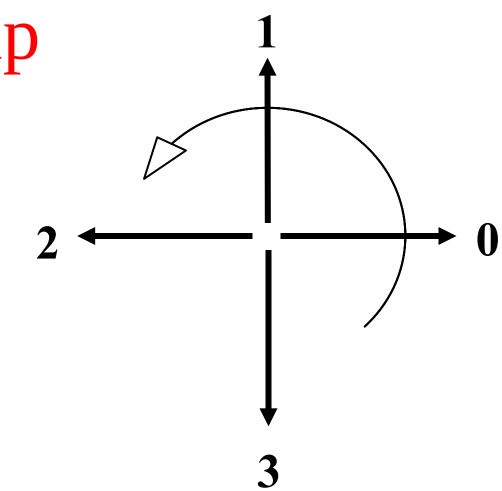


Chain code: **076666553321212** **766665533212120** **121207666655332**

Of *all possible* chain codes, 076666553321212 is the minimum integer

Normalisation (2)

- The chain code can also be made invariant to rotations of *multiple of 90 (4-driection) or 45 (8-direction) degrees*:
  - Find the *first difference* (or called derivative) of a chain code, which is obtained by treating the code as a circular sequence and counting the changes in directions between two neighbouring digits of the chain code
  - E.g., Suppose 10103322 is a chain code (using 4-directions)
    - From **1 to 0**: *3 changes* in direction (counter-clockwise),
    - From **0 to 1**: *1 change*,
    - From **1 to 0**: *3 changes*
    - ....
    - From the last digit **2** to the first digit 1: *3 changes. (some book ignore this digit)*
  - Therefore the code of first difference of 10103322 is 31330303



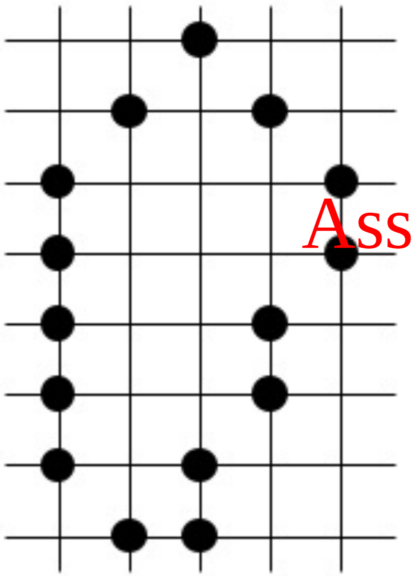
Shape Number

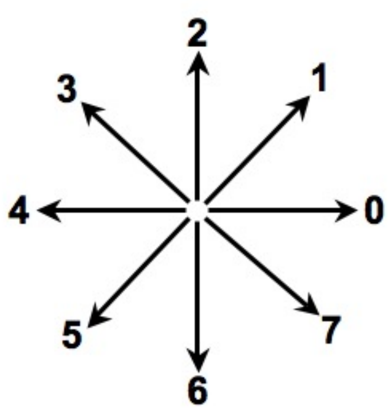
- A raw chain code is first made invariant to start point by finding the minimum of all possible chain codes (this step can be omitted);
- Then, the first difference of the minimum chain code is calculated;
- find the minimum integer of the first difference code – this number, invariant to both start point and rotations, is the shape descriptor, or called the shape number of the shape.
- In principle, the shape number is invariant to rotation (multiple of 90 or 45 degrees) and start point. In practice, due to the accuracy of the boundary discretisation, for a given shape, the shape number may vary slightly.

Cont’d

- The process of normalisation (or shape number calculation) uses the minimum value twice:
  - Find the raw code and its first difference (if needed, find the minimum of the raw chain code to make it *invariant to start point*);
  - After finding the first difference, find the minimum integer of first difference. This makes the code *invariant to rotations* (of multiple of 45 or 90 degrees depending the number of directions used).
  - If first difference is calculated without normalising for start point, the difference between the last digit and the first digit must be included in the first difference code.

Example





Raw chain code: (Starting from the top point)

776565643222211

Minimum integer from the raw code :

117765656432222

First difference of the chain code (excluding the difference between the last and the first digit).

06077171677000

Shape number:

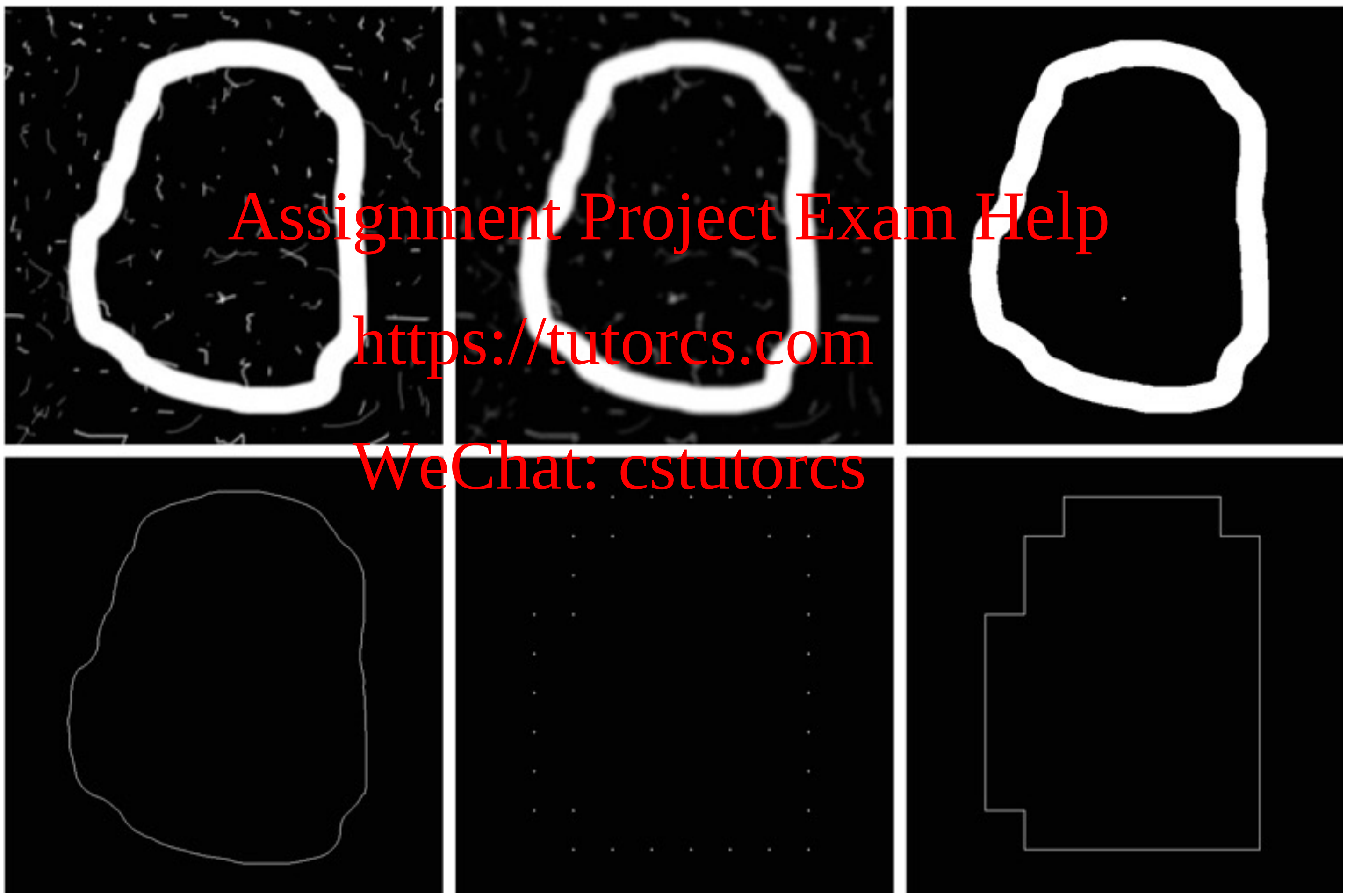
00006077171677

Get Chain Code in Matlab

- Smooth the image if it is noisy (it nearly always is);
- Convert the image into a binary image;
- Extract the external boundaries of shapes;
  - Matlab function: **`bwboundaries().`** This returns the boundaries of all shapes in a ***binary*** image.
- Down sample a boundary to reduce the number of boundary points (to make shorter codes);
  - Matlab function: **`bsubsamp()`** (A third party m-function)
- Produce the chain code, difference code, and shape number.
  - Matlab function: **`fchcode()`** (third party)

Example

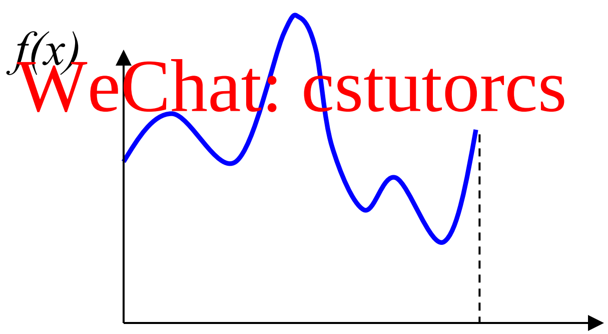
Original image | Smoothed image | Binary image



Boundary | Sampled Boundary | Connected sample points

Represent Shape by Its Components

- A less intuitive shape representation method is to decompose a general shape into some simple and well-known shapes (called the basis – the constituent components of shapes) and use the simple shapes to describe the general shape.



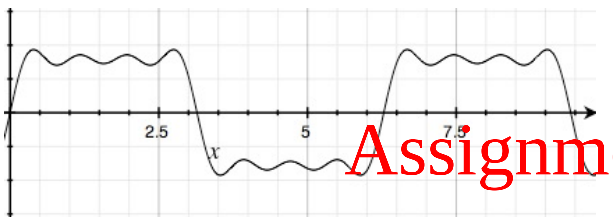
- What does this shape consist of?
  - There are different ways for finding them: Fourier transform and wavelet theory.

Shape Decompostion: An Example

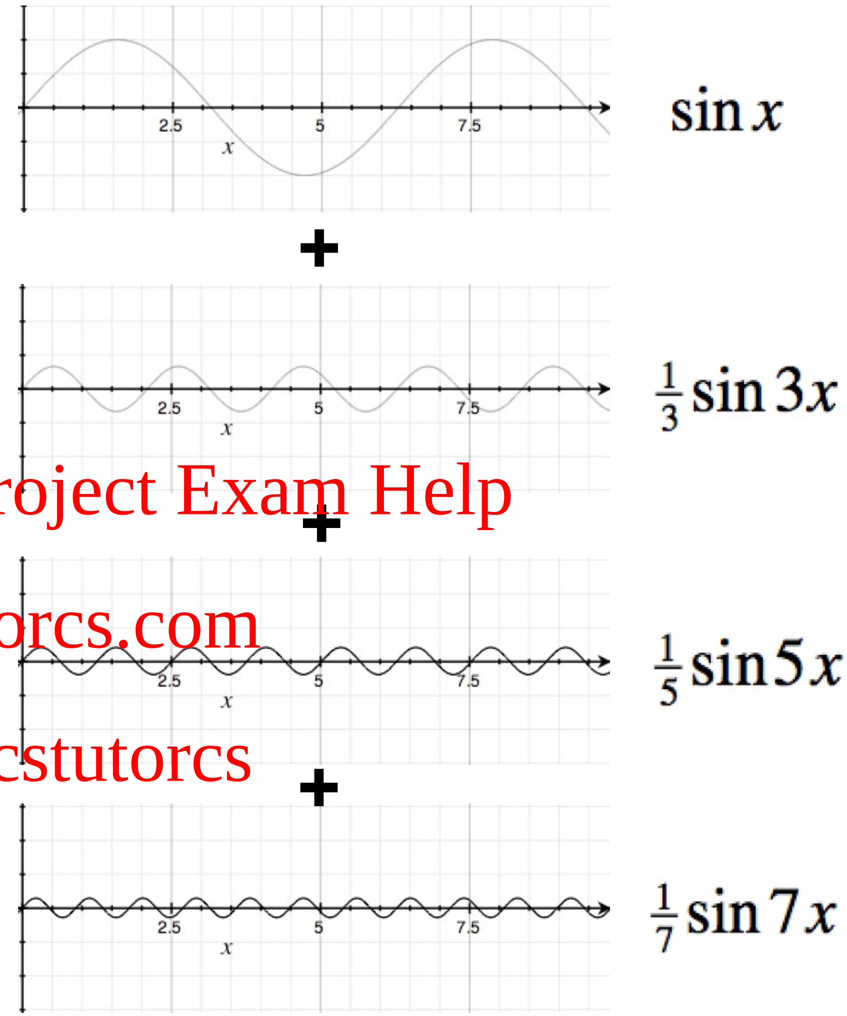
- A square wave (shape) has the decomposition

$$f(x) = \sin x + \tfrac{1}{3}\sin 3x + \tfrac{1}{5}\sin 5x + \tfrac{1}{7}\sin 7x + \tfrac{1}{9}\sin 9x + \cdots$$

$\sin x$, $\sin 3x$, $\sin 5x$, ... are well known sine curves/waves of periods 2π, $2\pi/3$, $2\pi/5$, ...

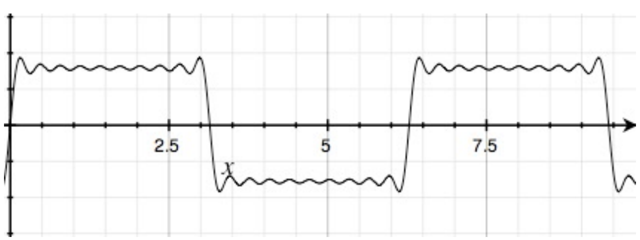


=



The result of the sum of the first 4 terms

The higher frequency components do not contribute much to the overall shape



The **sum** of first 10 terms.

More terms are taken in summing, the closer the sum gets to the square wave.

Cont’d

- The example shows that a square wave is the result of superposition (sum) of many sinusoidal waves of different frequencies.

$$f(x) = \sin x + \tfrac{1}{3}\sin 3x + \tfrac{1}{5}\sin 5x + \tfrac{1}{7}\sin 7x + \tfrac{1}{9}\sin 9x + \cdots$$

In General…

- Under certain conditions, given a function $f(x)$, we can always decompose it into the form

$$f(x) = const + a_1 \sin x + b_1 \cos x + a_2 \sin 2x + b_2 \cos 2x + \ldots$$
$$= const + \sum (a_n \sin nx + b_n \cos nx)$$

- Of course, for different shapes, the coefficients a_0, b_0, a_1, b_1, … a_n, b_n, will be different – some will be big, some will be small or zeros, e.g., for the square wave, all b_n are zeros.
- These coefficients control the weights of each of the trigonometric functions and therefore control the shapes – we can use them to describe shapes.

Find the Coefficients

- Therefore, in this approach shape description has become a problem of finding the coefficients for the trigonometric functions.
- Note that there are methods that use other functions, e.g., Wavelet functions (Haar wavelet), as the basis to decompose functions/curves. But such methods are beyond the scope of this module.

- Given a function, how can we find the coefficients for the trigonometric functions?

Fourier Transform (FT)

- The coefficients for a particular shape (function) can be found by Fourier Transform (the integral of the product of the function and the exponential function), and the coefficients found are called Fourier Transform of the function:

$$F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)e^{i\omega x}dx$$

- Conversely, if the Fourier transform (the coefficients) of a shape/function are known, one can reconstruct the original shape/function – this process is called Inverse Fourier Transform

$$f(x) = \int_{-\infty}^{\infty} F(\omega)e^{i\omega x}d\omega$$

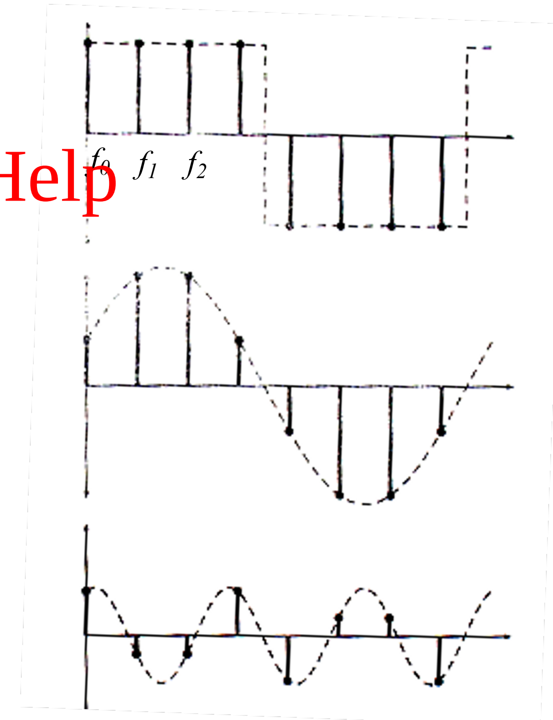
- This two formulas are called Fourier Transform pair.

DFT FFT

- To do the Fourier transform digitally, we first need to sample a continuous function into the discrete form, i.e., to represent it by samples

$$f(x) \Rightarrow [f_0, f_1, f_2, f_3, \ldots, f_{N-1}]$$

- When Fourier transform is applied to discrete functions (samples), it is referred to as having the Discrete Fourier Transform (DFT) of the function.
- There are very fast algorithms to compute DFT - they are called Fast Fourier Transforms (FFTs).



Applications of FFT

- Fast Fourier Transform is fundamentally important in modern technology.
- Fourier transform can be used to filter signals
  - First compute the Fourier Coefficients (complex numbers) of a (sampled) signal by FFT,
  - Then set some coefficients (associated with noises) to small or zero values to remove their effect in the signal, e.g.,
    - Remove high frequency noise: set the coefficients of high frequency components to zero
    - Remove noise of some specific frequencies: set the corresponding coefficients to zero
  - Then reconstruct the signal using the remaining coefficients by inverse FFT
- Signal and image compression: retaining and transmitting the coefficients of low frequency components

Image corrupted by sinusoidal noise

The white spots represent the frequencies of noise signals

Filter the transform of the image by masking out the transform of the noise

Reconstructed image

Fourier Descriptors

- When Fourier transform is used for describing boundary shapes, the resulted coefficients are called Fourier Descriptors.

- Advantages:
  - A few descriptors are enough to describe the gross shape.
  - They can be made invariant to basic transformation (rotation and scale) of the shapes.

Compute Fourier Descriptors

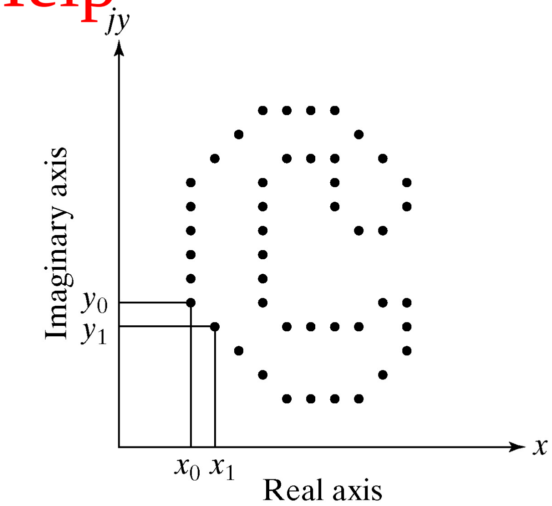
- Given a boundary, starting at an arbitrary point (x_0, y_0), traverse the boundary, say, in counterclockwise direction, then the boundary can be represented by the coordinates of the boundary points:

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{N-1}, y_{N-1})$$

- Convert the coordinates to complex numbers:

$$s(k)=x_k+iy_k \quad for\ k=0, 1, 2, \ldots, N-1$$

- Transform this series of complex numbers using FFT.
- The obtained Fourier coefficients (complex numbers) are the Fourier descriptors of the shape.
- Original shape can be constructed from a small numbers of the coefficients (descriptors).



Example

A human chromosome

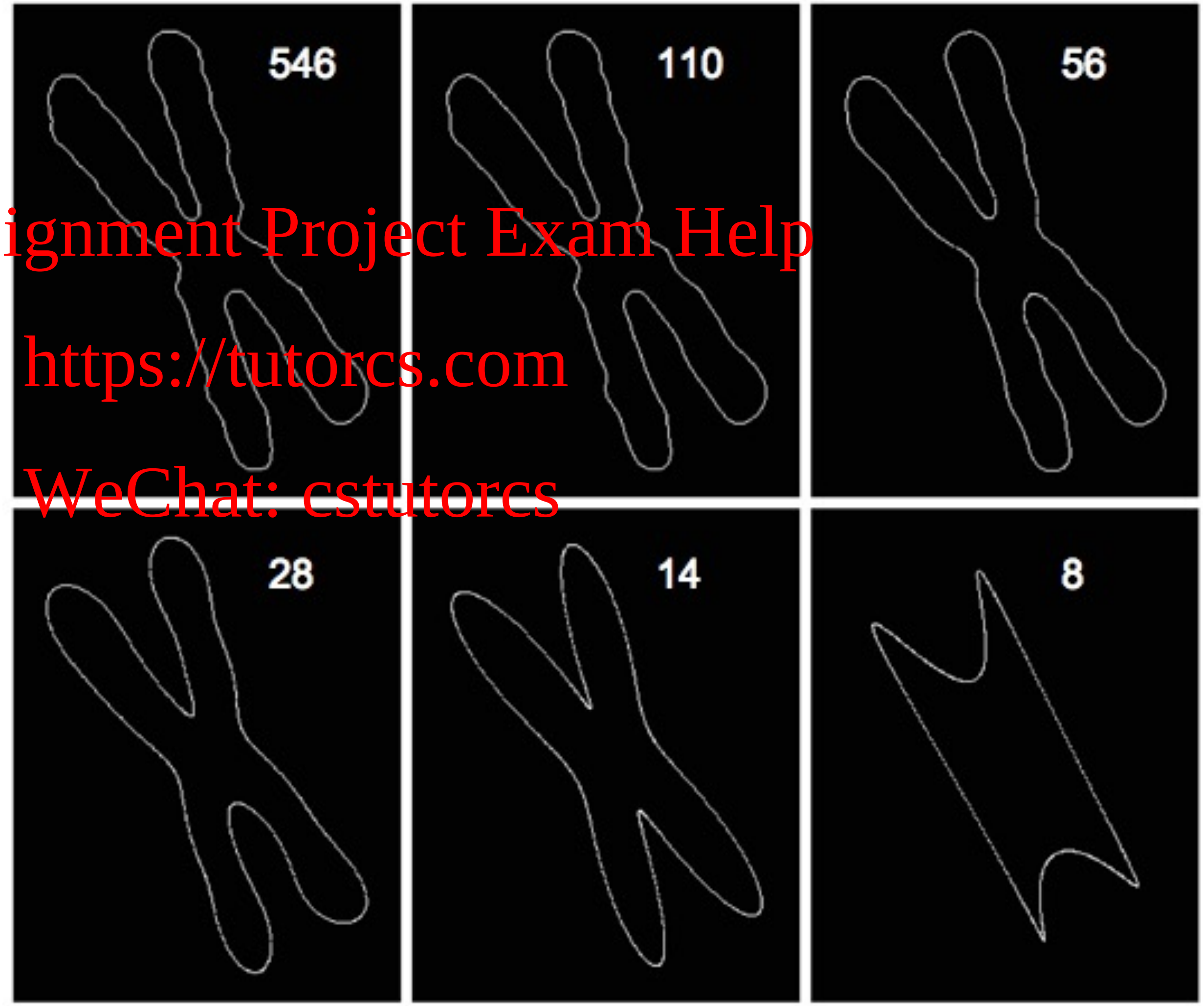
Binary image

Boundary (1090 points)

The coordinates of the boundary points (1090) are converted to complex numbers (1090). Then apply FFT to them.

Reconstructed Boundary

The numbers show the numbers of descriptors used in reconstruction



Matlab Implementation

- Functions `frdescp` and `ifrdescp` implement the transforms.
- `frdescp` takes boundary points (x and y pairs of type double) and pre-processes the boundary data (convert them to complex numbers) and then calls the standard FFT algorithms `fft()` of IPT.
- `ifrdescp(`*`descp, n`*`)` takes two inputs:
  - `descp` is the Fourier descriptors, an array of complex numbers,
  - `n:` is the number of descriptors you want to keep/use for reconstruction. The default value of n is the length of `descp`, i.e., all the descriptors.

Summary of Descriptors

- Most descriptors works well only in 2D
  - They work well if the shape is simple

- There is no easy solution in 3D
  - The problem of self-occlusion
  - Perspective projection
  - Hard to make the representation rotation invariant
  - Still an active research area

Further Readings

- Shapiro, L.G., Stockman, G.C., Computer Vision, Prentice-Hall, 2001, ISBN 0-13-030796-3
  - Section 10.2 for Freeman chain code
  - Section 5.11 for Fourier analysis