COMP3005/Computer Vision

5

Shape description and modelling

Summary

Basic shape description involves extracting characteristic features that describe the *shape* of a connected component. Shape descriptors can be used together with machine learning, or manually defined models to classify shapes into different classes; a common example of this is classifying shapes into letters of the alphabet in order to achieve optical character recognition. Statistical shape models describe how the shape of an object (or set of objects) can change (through internal movement and/or through changes in pose). In addition to describing shapes, statistical shape models can be used to find instances of a shape in an image.

Key points

Simple scalar features

- The following features are simple, single number descriptions of a shape (specifically a *connected component*):
  - **Area**: the number of pixels in a component
  - **Perimeter length**: the length around the outside of the component
    - **Inner border**: pixels within the component forming its edge
    - **Outer border**: pixels around the outside the component
    - Perimeter can be approximated as:

      $$P(S) = \sum_j \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}$$

      where x_j and y_j correspond to the x and y coordinates of (consecutive) pixels on the border.
  - **Compactness** is the ratio of the area to the perimeter squared:

    $$C(s) = \frac{4\pi A(s)}{(P(s))^2}$$

    the 4π factor normalizes the compactness so that a circle is the most compact shape possible (area of a circle = $\frac{(P(s))^2}{4\pi}$), with a compactness of 1.
    - *Note many authors define compactness in slightly different ways, although the concept is the same!*
  - **Irregularity** or **dispersion** can be measured as the ratio of major chord length to area. A simple way to compute it is as follows:

    $$I(s) = \frac{\pi \max\left((x_j - \bar{x})^2 + (y_j - \bar{y})^2\right)}{A(s)}$$

    where $\bar{x}$ and $\bar{y}$ correspond to the x and y coordinates of the centroid (i.e. the mean of the x and y positions of all pixels in the component).
    - Another alternative is to represent dispersion as the ratio of the smallest circle that encompasses the shape to the largest circle that can fit inside the shape:

      $$IR(s) = \frac{\max\left(\sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2}\right)}{\min\left(\sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2}\right)}$$

Written by Jonathon Hare, February 19, 2014. Send any comments/bugs/typos to jsh2@ecs.soton.ac.uk.

Moments

- Moments describe the distribution of pixels in a shape.
  - Moments can be computed for any grey-level image. For the purposes of describing shape, we'll just focus on moments of a connected component.
  - Standard **two-dimensional Cartesian moment** of an image, with order p and q is defined as:

$$m_{pq} = \sum_x \sum_y x^p y^q \, I(x,y) \Delta A$$

  - In the case of a connected component, this simplifies to:

$$m_{pq} = \sum_i x_i^p y_i^q$$

    - The zero order moment of a connected component m_{00} is just the area of the component. The centre of mass is (centroid):

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}$$

  - Standard 2d Cartesian moments can be used as descriptors of a shape
    - Different shapes have different moments
    - But, not invariant to: translation, rotation, scaling, …
  - **Central moments** are translation invariant as they compute moments about the centroid of the component:

$$\mu_{pq} = \sum_i (x_i - \bar{x})^p (y_i - \bar{y})^q$$

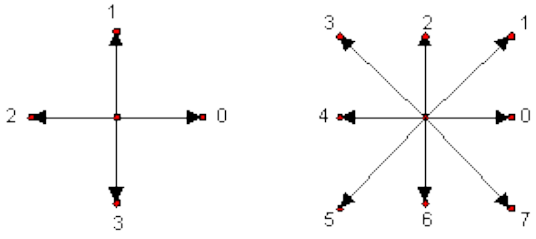
    - Clearly, the zero order centralised moment is the area
    - The two first-order central moments are (μ_{10} and μ_{01}) both zero!
      - They have no description power
    - Higher order central moments are descriptive however
  - **Normalised central moments** are both translation and scale invariant:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \text{ where } \gamma = \frac{(p+q)}{2} + 1$$

  - **The Hu Moments** (named after the inventor) are a set of 7 scale, rotation and translation invariant moments created from combinations of the normalised central moments.
  - It is possible to go even further: there are actually moments that are *affine invariant!*

Chain codes

- Chain codes are a simple way of encoding the boundary of an object. Simply computed by walking around the boundary and noting the direction of each step (either in 4 or 8 directions).



  - By rotating the chain code sequence so that it forms the smallest possible integer the code can be made invariant to the point on the shape you start from.
  - Rotation invariance can be achieved by storing a differential code (i.e. storing the differences in consecutive numbers, rather than the numbers themselves).
  - Scale invariance is (in theory) possible to achieve by resampling the shape to a fixed size.
  - Can be used to compute area, perimeter and moments directly!
    - Perimeter for an 8-connected chain code is N(even numbers in chain code) + $\sqrt{2}$ * N(odd numbers in chain code)
  - Practically speaking, they are not so good for shape matching due to noise, resampling issues and problems of generating good similarity/distance metrics.

Written by Jonathon Hare, February 19, 2014. Send any comments/bugs/typos to jsh2@ecs.soton.ac.uk.

Fourier descriptors

- The Fourier transform can be used to encode shape information by decomposing the boundary into a (small) set of frequency components.
    - There are two main steps to consider:
        - Defining a representation of a curve (the boundary)
        - Expanding the representation using Fourier theory.
    - By choosing these steps carefully it is possible to create rotation, translation and scale invariant boundary descriptions that can be used for recognition, etc.

Describing multiple shapes: Region Adjacency Graphs

- Region adjacency graphs (actually they're just trees) can be used to describe the layout of multiple connected components relative to each other.
    - Nodes in the graph correspond to components
    - Nodes are connected if they share a border
- Invariant to distortion (including rotation, scale, translation, as well as non-linear transformations).
- Not invariant to occlusion though.
- Good for creating markers like QR codes, which are more natural
    - Recognition just involves finding a sub-tree structure that matches a set of known tree structures, so is very fast.

Point Distribution Models

- A few lectures ago, we saw how PCA could be applied to face images in the classic Eigenfaces algorithm. A Point Distribution Model (PDM) applies a similar process to a set of points representing a shape.
    - Taking the example of faces:
        - Sets of corresponding 2D points (i.e. points corresponding to the same physical feature [e.g. the tip of the nose]) are manually created from a set of N training face images (typically containing different facial expressions and poses).
            - The number of points is fixed at M; all M points must exist on all images!
        - An iterative process called Generalized Procrustes Analysis is used to align all the point sets from each image (so they all have the same size and rotation and are centred about the origin).
        - The mean shape is created by considering the average position of all of the aligned points that correspond to each other.
        - An Nx$2M$ shape matrix is formed such that each row corresponds to one of the training images, and each pair of columns corresponds to the x and y coordinates of an aligned point. Corresponding points across the images are stored in the same pair of columns.
            - A good way of thinking about this, is that each column stores information on how the x or y ordinate of a point on a face can change.
        - PCA is applied to this matrix to generate a low dimensional basis that fundamentally can describe different facial expressions and poses.
            - Any shape vector can be projected against this matrix to generate a low-dimensional description (which encodes pose and expression).
            - A low dimensional description vector can be reconstructed back into a set of (x, y) coordinate pairs representing a face.

Active shape models/Active appearance models/Constrained Local Models

- Active shape models (ASMs) and Constrained Local Models (CLMs) all take a point distribution model a step further, and incorporate models of what the image should look like around each point in the PDM.

Written by Jonathon Hare, February 19, 2014. Send any comments/bugs/typos to jsh2@ecs.soton.ac.uk.

    - In the simplest case, this is just a small patch of pixels (e.g. based on the average across the training images) about each point – this is a *template*.
    - The addition of this data allows the model to be fitted to an unseen image using an algorithm that iteratively updates the points to their best position (e.g. using template matching in a small window around the point), and updating the points to positions considered to be plausible by the PDM, by projecting them into the low dimensional space and then reconstructing them in the original space.
        - Essentially each point tries to move to its local optimum, whilst the PDM constrains all the points to still looking like a face!
- Active Appearance Models (AAMs) do the same thing as ASMs and CLMs, but instead of local features around each point, they try to jointly optimise the global appearance of the face (based on an Eigenfaces modal) against the PDM.

Further reading

- Mark's book (third edition), covers shape description in Chapter 7 (including detailed information on Fourier descriptors). Another good reference is Sonka, Hlavac and Boyle "Image Processing, Analysis and Machine Vision" (chapter 6 in the second edition).
- Active shape/appearance models are described in detail in this document by their inventor (don't worry about the details; just get an understanding of the idea): http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/Models/app_models.pdf
- The constrained local model demonstrated in the lecture implements this paper (it's complex; don't worry about the details): http://www.ri.cmu.edu/pub_files/2009/9/CameraReady-6.pdf

Practical exercises

- Create some connected component objects using OpenIMAJ (see chapter 3 of the tutorial if you haven't done it already). Explore some of the methods on the `ConnectedComponent` object for creating simple shape descriptions. Once you've done that, have a look at the classes in the `org.openimaj.image.connectedcomponent.proc` package for some more advanced shape features.
- You too can play with the CLM model! Try hooking up an instance of the `org.openimaj.image.processing.face.tracking.clm.CLMFaceTracker` to video from a webcam. The `drawModel` method will help you draw the model on your video frames.

Written by Jonathon Hare, February 19, 2014. Send any comments/bugs/typos to jsh2@ecs.soton.ac.uk.