Chapter 8

Object Recognition

Object Recognition is an important area of Computer Vision in which algorithms segment or classify objects based on characteristics of their images. The process may use either raw images or transformed images. This topic overlaps with Artificial Intelligence, but it has specific approaches to deal with feature extraction and training. Object recognition is an important step towards Image Understanding.

In section 8.1 feature extraction is presented, and the difference between invariant features and non-invariant features is discussed. In section 8.2 a real-time approach for object detection is presented.

8.1 Feature Extraction

Feature extraction is the process of computing a limited number of features that describes a large set of data. Features are characterised by performance, accuracy and discrimination powers.

Performance is limited by the complexity of the feature extraction method. For example, extraction methods based on FFTs are generally slow, especially if they involve multiresolution analysis. Example of fast methods include Haar-like features.

The accuracy of feature extraction is affected by factors such as lighting condition, shades, rotation, translation and articulation of the objects. For example, extracting features from images of Latin characters can be relatively simple. Suppose one has binary images of a certain fixed size and can reliably count the number of colour changes in each line and column (see figure 8.1. These features can characterise each object uniquely, even if different fonts are used. However, if the characters were rotated, then the feature extraction might yield a different set of numbers.

8.1.1 Invariant Features

Researchers have been trying to find features that are invariant to the geometric transformations. The most famous of these sets of features is called Hu's Moment Invariants ([1]), often used in OCR (optical character recognition). Although Hu's features can be extended to higher orders, in practice they are limited due to noise effects. Hu's moment invariants are invariant to translation, scaling, and rotation, but are not very robust against other transformations.

Given a digital image $i(x, y)$, the 2D moment of order $(p + q)$ is:

$$m_{pq} = \sum_x \sum_y x^p y^q i(x, y) \tag{8.1}$$

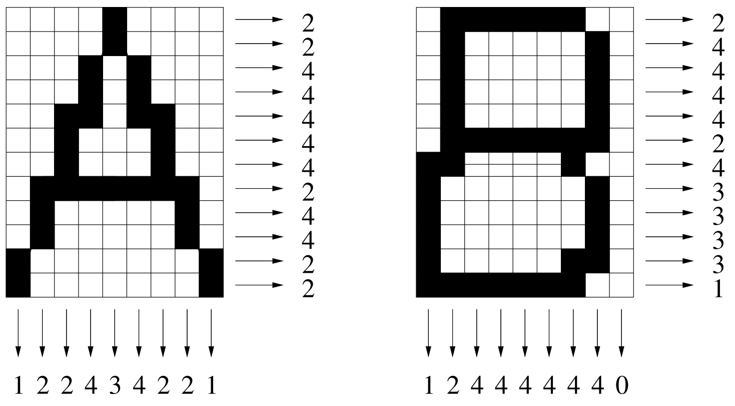

Figure 8.1: Simple feature extraction for binary images of Latin characters.

For any order (p+q), each element can be precomputed by multiplying the pixel value by its position. It is trivial to create SATs for 2D moments of any order. 2D moments are non-invariant, but they are the basis for Hu's equations. The values $\bar{x}$ and $\bar{y}$ are given by:

$$\bar{x} = \frac{m_{10}}{m_{00}} \tag{8.2}$$

and

$$\bar{y} = \frac{m_{01}}{m_{00}} \tag{8.3}$$

The central moment μ_{pq} is defined by:

$$\mu_{pq} = \sum_x \sum_y (\bar{x} - x)^p (\bar{y} - y)^q i(x, y) \tag{8.4}$$

And the normalised central moment η_{pq} is given by:

$$\eta_{pq} = \frac{\mu_{pq}}{{\mu_{00}}^{\gamma}} \tag{8.5}$$

Where $\gamma = \frac{p+q+2}{2}$

The following equations are for the seven 2-D Moment Invariants proposed by Hu. They are invariant to translation, scaling, rotation, and mirroring:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{8.6}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{8.7}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \tag{8.8}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \tag{8.9}$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2] \\ + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{8.10}$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - ((\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{8.11}$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2] \\ + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{8.12}$$

Flusser [2] recommended that Hu's moments ϕ_2 and ϕ_3 should not be used in 2-D object recognition, as they are dependent on the others. OpenCV has an implementation of Hu's invariants. One example of application for real-time moment invariants can be seen in [3].

8.2 Real-time Object Detection

The search for a method that can find faces reliably and quickly has been the focus of attention for many years. Only recently such methods became available. There are other methods that are accurate, but they usually take at least few seconds for each frame to be examined.

In 2001 Viola and Jones ([4]) published a paper on real-time object recognition. In this paper they derived a method that uses known data structures and algorithms combined in an original way, and showed impressive results for face detection. [1]

The method makes three new contributions to the field of real-time object detection:

- It uses a simple feature called Haar-like feature that can be computed rapidly through the use of integral images.
- The training method uses a customised version of an algorithm called *Adaboost*, which produces classifiers using Haar-like features. These features are selected with a minimum error criteria.
- In order to speedup the search, classifiers are organised in cascades.

8.2.1 Haar-like Features

These features are based on the idea of Haar functions used in wavelets. The sum of pixels of rectangular areas are compared to the sum of pixels of adjacent areas. These are usually too inaccurate to be used in object recognition. However, previous work done by other researchers have demonstrated that many of these features together can represent patterns in an image.

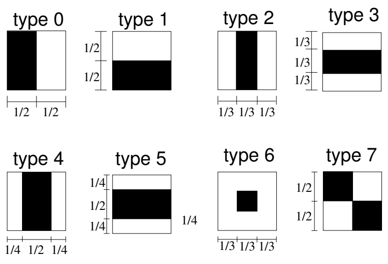

Figure 8.2: Examples of Haar-like Features.

The Haar-like features are said to over-represent the objects, as they are redundant and more numerous than the image pixels.

In order to compute the features rapidly, Viola and Jones used a data structure called integral images (also called Summed-area Tables (SAT)). The integral images are computed once for the whole frame, and takes only 4 lookups to get rectangular sums of pixels from any position, at any scale. Given an image $i(x_i, y_i)$, the integral image $I(x, y)$ is:

$$I(x,y) = \sum_{x \leq x_i, y \leq y_i} i(x_i, y_i) \tag{8.13}$$

Now the sum of rectangular areas over the image can be computed with 4 look-ups (figure 8.3).

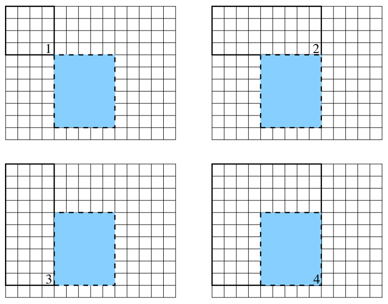

Figure 8.3: The integral image approach.

In order to compute a square area, four lookups in the integral image are needed:

$$Area = pt_4 - pt_2 - pt_3 + pt1 \tag{8.14}$$

[1]face detection differs from face recognition for the fact that the former finds any face, while the latter finds a specific individual.

8.2.2 Training

After extracting features from the images, one needs to make some sense out of all that data. If there is a simple correlation between the feature values and the class of objects, the classification is a simple problem to solve. However, that is rarely the case. One needs to use AI techniques, such as training algorithms, to correlate the data with a certain object. In this section we present an algorithm called AdaBoost.

Adaboost was proposed in the late 90's based on boosting techniques ([5]). The algorithm for the discrete version can be seen below. Adaboost is simple to implement, can cope with large dimensional space and it is numerically stable. The idea behind Adaboost is that weak classifiers can be "boosted", i.e., a set of inaccurate classifiers can compose a strong classifier.

Algorithm 6 ***AdaBoost***$((value, class), T)$

```
The input is a set of training examples $(x_1, y_1), (x_2, y_2), ...(x_2, y_2)$ where $x_i$ is a feature $\in$ X and $y_i$ is a class $\in \{-1, +1\}$, and a number of rounds $T$. It outputs a linear combination of weak classifiers $h_t(x_i)$, each a factor $\alpha_t$.
Initialise weights for each element $D_1(i) = 1/(n)$ where $n$ is the number of elements
For $t = 1, 2...T$:
    1. Train a weak classifier $h_t$ (e.g. a simple threshold) using the weights in $D_1$ so that $h_t \in \{-1, +1\}$
    2. Compute the error associated with the weak classifier:
         $error_t = \sum_i (D_t(i) h_t(x_i) y_i)$ (the sum of the elements' weight that are incorrectly classified)
    3. Compute $\alpha_t = 0.5\ ln(\frac{(1-error_t)}{error_t})$
    4. Update the weights $D_{t+1}(i)$ such as $D_{t+1}$ is a new distribution:
         $D_{t+1}(i) = D_t(i)\ exp(-\alpha_t y_i h_t(x_i))$
              (the weights decrease if the element is classified correctly or increase if it is classified incorrectly)
         normalise the weights $D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_i D_{t+1}(i)}$
    5. After $T$ rounds the final classifier is: $H(x) = sign(\sum_t \alpha_t h_t(x))$
```

Figure 8.4 shows an example for two dimensions. The features in this case relate to the colour of the pixels. A simple threshold is used for each weak classifier. The weak classifiers are better than 50% (better than chance), but they are very innacurate. By combining them linearly it is possible to define regions of the space that contain either positive or negative examples of the object. Eventually the training can achieve 0% error rate. In practice, it is difficult to achieve such low training errors, specially if the sample sets are large.

8.2.3 Cascade of Classifiers

Adaboost can generate a single classifier, composed by several hundred weak classifiers. However, Viola and Jones noticed that they can improve the performance without any loss in accuracy if they split this monolithic classifier in several pieces (figure 8.5). Each *stage* (or *layer*) eliminates a portion of the sub-windows that can not contain any object, while preserving most of the sub-windows that might contain an object.

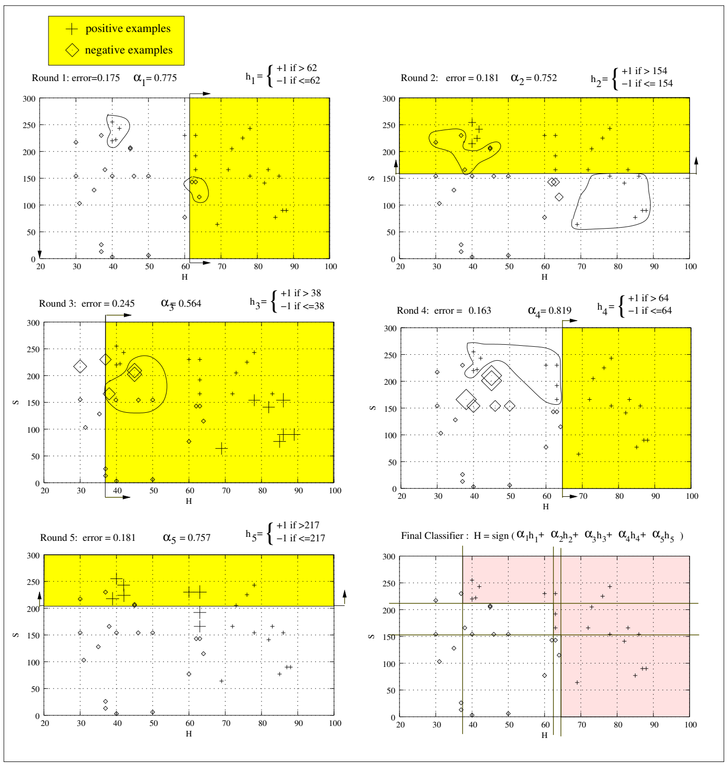

Figure 8.4: Training 2 dimensional feature space using Adaboost.

In practice each step is constructed by training with Adaboost until a maximum training error is reached. To start the next step training, new negative examples are chosen. These new negative examples need to be tested against the current state of the classifier, i.e., it should be a false positive. In this sense the training of subsequent steps is harder (computationally more expensive than the previous steps). Training may take days or weeks, but when the classifier is produced the method can detect the object in real-time.

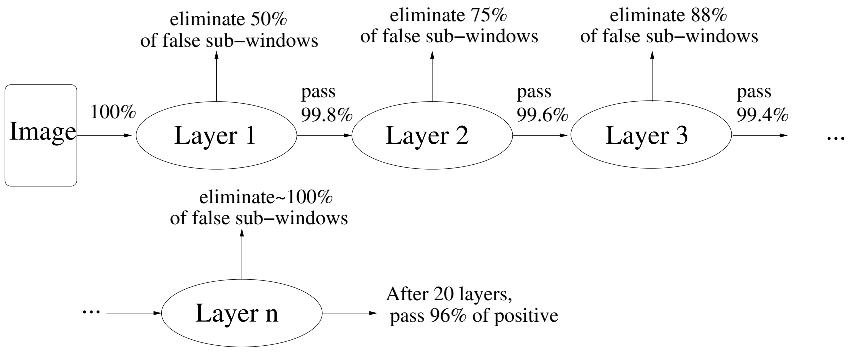

Image
100%
Layer 1
eliminate 50% of false sub−windows
pass 99.8%
Layer 2
eliminate 75% of false sub−windows
pass 99.6%
Layer 3
eliminate 88% of false sub−windows
pass 99.4%
...
...
Layer n
eliminate~100% of false sub−windows
After 20 layers, pass 96% of positive

Figure 8.5: A classifier organised in cascades

8.2.4 Rotated Haar-like Features

Extensive experiments with rotated Haar-like Features where carried out at Massey, with some interesting results. The limitations are the representation of the Haar-like Features, as it cannot be generalised to an arbitrary angle. An attempt to combine two features to get arbitrary angles was unsuccessful, the accuracy of the rotated Haar-like features are not enough to be used in classifiers. However, these experiments were concluded with a simple conversion method in which the user can create new rotated classifiers without the need to retrain them, as long as the new classifiers are at angles that are multiple of 45 degrees. Classification of rotated faces and rotated gestures can be achieved using parallel cascade classifiers that were converted using such method (see for example [6] and [7]).

8.3 Reading

Read chapter 12 of Gonzalez & Woods. Extra reading: [8].

8.4 Exercises

1. Write a program that can compute similarity between two histograms at a certain size. Using a stored histogram of an object, search for the same object in different images.

2. Use a colour histogram (HSV) and compare the accuracy.

3. Run the opencv demo code for detecting a human face. Try to rotate the camera to assess at which angle the face is no longer detected. Test other classifiers available on OpenCV to compare the accuracy with the face detector.

Bibliography

[1] M.-K. Hu, “Visual pattern recognition by moment invariants,” *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, 1962.

[2] J. Flusser, “On the independence of rotation moment invariants,” *Pattern Recognition*, vol. 33, pp. 1405–1410, 2000.

[3] A. L. C. Barczak and M. J. Johnson, “A new rapid feature extraction method for computer vision based on moments,” in *International Conference in Image and Vision Computing NZ (IVCNZ 2006)*, (Auckland, NZ), pp. 395–400, November 2006.

[4] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *CVPR01*, (Kauai, HI), pp. I:511–518, IEEE, December 2001.

[5] Y. Freund and R. E. Schapire, “A short introduction to boosting,” *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[6] A. L. C. Barczak, “Toward an efficient implementation of a rotation invariant detector using haar-like features,” in *IVCNZ05*, (Dunedin, NZ), pp. 31–36, 2005.

[7] A. L. C. Barczak, F. Dadgostar, and C. H. Messom, “Real-time hand tracking based on non-invariant features,” in *IMTC2005*, (Ottawa, Canada), pp. 2192–2199, 2005.

[8] P. Viola and M. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, pp. 137–154, May 2004.