Applied Computer Vision

David Vernon
Carnegie Mellon University Africa

vernon@cmu.edu
www.vernon.eu

Lecture 15

Object Recognition

Statistical Pattern Recognition:
2D Shape Features & Classification

Image Analysis

- Reminder: automatically extracting useful information from an image
- We can also classify the types of analysis we wish to perform according to function
  - Inspection

    Is the visual appearance of objects as it should be?
  - Location

    requires the specification of both position and orientation in either 2D or 3D
    - image frame of reference (pixels)
    - real world frame of reference (e.g. millimetres) … calibration required
  - Identification of object type

Statistical Pattern Recognition

- Find objects within the image and identify or classify those objects

- Central assumption:
  - the image depicts one or more objects
  - each object belongs to one of several distinct and exclusive predetermined classes

- We know what objects exist and an object can only have one particular type or label

Statistical Pattern Recognition

Three components of this type of pattern recognition process:

- an object isolation module
  - Produces a representation of the object (segmentation)
- a feature extraction module
  - Abstracts one or more characteristic features and produces a feature vector
  - Selection of features is crucial
- a classification module
  - The feature vector is used by the classification module to identify and label each object

Features

Features should be

- Independent

  a change in one feature should not change significantly the value of another feature

- Discriminatory

  Each feature should have a significantly different value for each different object

- Reliable

  Features should have the same value for all objects in the same class/group

- The computational complexity of pattern recognition increases rapidly as the number of features increases

- Hence it is desirable to use the fewest number of features possible, while ensuring a minimal number of errors

Features

Simple feature extraction

- Many features are either based on the size of the object or on its shape
- Area of the object
  - simply the number of pixels comprising the object multiplies by the area of a single pixel (frequently assumed to be a single unit)
  - can also be computed from the boundary contour (see later)
- Integrated Optical Density (IOD)
  - the area multiplied by the average grey level of the object
  - provides a measure of the 'weight' of the object, where the pixel grey-level encodes the weight per unit area

Features

Simple feature extraction

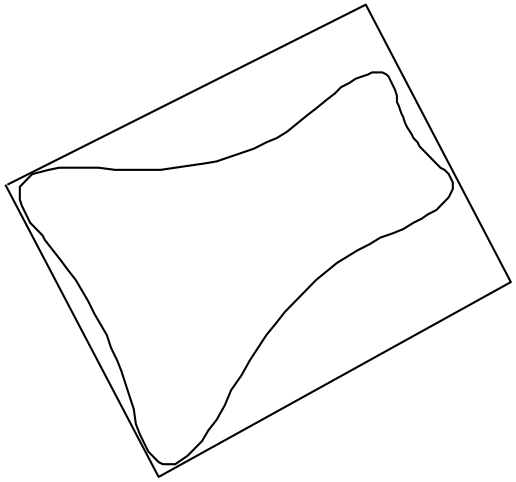
- The length and the width of an object describe
- If its orientation is not known
  - we may have to first compute its orientation before evaluating the minimum and maximum extent of its boundary
- These measures should always be made with respect to some rotation-invariant datum line in the object, *e.g.*, its major or minor axis

Features

Simple feature extraction

Minimum Bounding Rectangle

- The smallest rectangle which can completely enclose the object
- The main axis of this rectangle is the principle axis of the object
- Hence, the dimensions of the minimum bounding rectangle correspond to the features of length and width

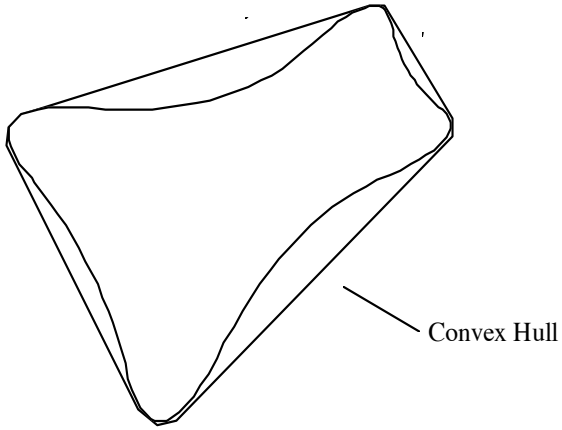


Features

Simple feature extraction

Convex Hull

- The smallest convex boundary which can completely enclose the object



Features

Simple feature extraction

- The distance around the perimeter of the object
- Depending on how the object is represented, it can be quite trivial to compute the length of the perimeter
- This makes it an attractive feature for industrial vision applications

Features

Simple feature extraction

Rectangularity

$$R = \frac{A_{\text{object}}}{A_{\text{min. bound. rectangle}}}$$

- Minimum value of 1 for a perfect rectangular shape
- Tends toward zero for thin curvy objects

Rectangularity : Aspect Ratio

$$\text{Aspect Ratio} = \frac{W_{\text{min. bounding rectangle}}}{L_{\text{min. bounding rectangle}}}$$

Features

Simple feature extraction

Elongatedness

$$\frac{A_{object}}{(2d)^2}$$

- Ratio of object area to square of its "thickness" d

- "Thickness" can be estimated by
  - the number of iterations of an erosion operator to remove the object
  - the number of iterations of a thinning operator

Features

Simple feature extraction

Circularity

$$C = \frac{A_{object}}{P^2_{object}}$$

- Maximum value for discs
- Tends toward zero for irregular shapes with ragged boundaries

Features

Shape Descriptors

Types of shape descriptors (Pavlidis 1978)

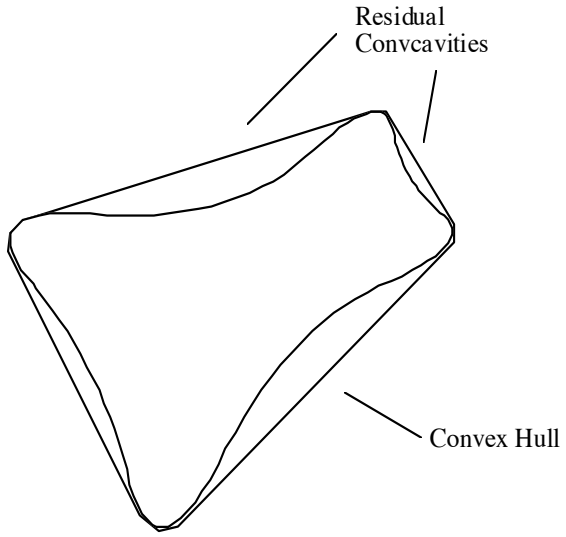
- Based on external or internal properties of the shape
  - boundary vs.
  - region properties
- Based on scalar transform techniques or on space domain techniques
  - vectors of scalar features vs.
  - relational descriptors of shape features
- 4 types

1. External Scalar Transform techniques: features of the shape boundary
2. Internal Scalar Transform techniques: features of the shape region
3. External Space Domain techniques: spatial organisation of the shape boundary
4. Internal Space Domain techniques: spatial organisation of the shape

Features

External Scalar Transform Shape Descriptors

- Perimeter length
- Ratio of the major to minor axis of the minimal bounding rectangle
- Circularity
- Number and size of residual concavities lying within the bounds of the shape's convex hull
- ...



Features

Internal Scalar Transform Shape Descriptors

- Method of Moments
- The standard two-dimensional moments m_{uv} of an image intensity function $g(x, y)$

$$m_{uv} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)x^u y^u dxdy \qquad u, v = 0, 1, 2, 3\ldots$$

$$m_{uv} = \sum_x \sum_y g(x,y)x^u y^u \qquad u, v = 0, 1, 2, 3\ldots$$

summed over the entire sub-image within which the shape lies

Features

Internal Scalar Transform Shape Descriptors

- Method of Moments

- However, these moments will vary for a given shape depending on where the shape is positioned, *i.e.*, they are position-dependent

- Instead, use the central moments

$$m_{uv} = \sum_x \sum_y g(x,y)\left(x - \bar{x}\right)^u \left(y - \bar{y}\right)^u$$

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

i.e. the coordinates of the centroid of the shape

- This renders the moments position invariant

Features

Internal Scalar Transform Shape Descriptors

- Method of Moments

- Assuming that the intensity function $g(x, y)$ has a value of 1 everywhere in the object (*i.e.* we are dealing with a simple segmented binary image), the computation of m_{00} is simply a summation yielding the total number of pixels within the shape

- If we also assume that a pixel is one unit area, then m_{00} is equivalent to the area of the shape

- Similarly, m_{10} is effectively the summation of all the x co-ordinates of pixels in the shape and m_{01} is the summation of all the y co-ordinates of pixels in the shape; hence

  m_{10}/m_{00} is the average x co-ordinate

  m_{01}/m_{00} is the average y co-ordinate

  i.e. the co-ordinates of the centroid.

Features

Internal Scalar Transform Shape Descriptors

Central Moments

$$\mu_{00} = m_{00}$$
$$\mu_{10} = 0$$
$$\mu_{01} = 0$$
$$\mu_{20} = m_{20} - \bar{x}m_{10}$$
$$\mu_{02} = m_{02} - \bar{y}m_{01}$$
$$\mu_{11} = m_{11} - \bar{y}m_{10}$$
$$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2 m_{10}$$
$$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2 m_{01}$$
$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{01} + 2\bar{y}^2 m_{10}$$
$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}$$

Features

Internal Scalar Transform Shape Descriptors

Normalized Central Moments

$$\eta_{ij} = \frac{\mu_{ij}}{\left(\mu_{00}\right)^{k}}$$

where $k = ((i + j) / 2) + 1 \quad | \, i + j >= 2$

Features

Internal Scalar Transform Shape Descriptors

Moment Invariants

- Linear combinations of normalised central moments
- More frequently used for shape description because they generate values which are invariant with position, orientation and scale changes
- Also known as Hu moment invariants or Hu moments

Features

Internal Scalar Transform Shape Descriptors

Moment Invariants

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

$$\phi_2 = \left(\eta_{20} + \eta_{02}\right)^2 + 4\eta_{11}^2$$

$$\phi_3 = \left(\eta_{30} + 3\eta_{12}\right)^2 + \left(3\eta_{21} + \eta_{03}\right)^2$$

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

$$\phi_5 = \left(\eta_{30} + 3\eta_{12}\right)\left(\eta_{30} + \eta_{12}\right)\left\{\left(\eta_{30} + \eta_{12}\right)^2 - 3\left(\eta_{21} + \eta_{03}\right)^2\right\} +$$
$$\left(3\eta_{21} - \eta_{03}\right)\left(\eta_{21} + \eta_{03}\right)\left\{3\left(\eta_{30} + \eta_{12}\right)^2 - \left(\eta_{21} + \eta_{03}\right)^2\right\}$$

$$\phi_6 = \left(\eta_{20} + 3\eta_{12}\right)\left\{\left(\eta_{30} + \eta_{12}\right) - \left(\eta_{21} + \eta_{03}\right)^2\right\}$$
$$+ 4\eta_{11}\left(\eta_{30} - \eta_{12}\right)\left(\eta_{21} + \eta_{03}\right)$$

$$\phi_6 = \left(3\eta_{21} - \eta_{03}\right)\left(\eta_{30} + \eta_{12}\right)\left\{\left(\eta_{30} + \eta_{12}\right)^2 - 3\left(\eta_{21} + \eta_{03}\right)^2\right\}$$
$$- \left(3\eta_{12} - \eta_{30}\right)\left(\eta_{21} + \eta_{03}\right)\left\{3\left(\eta_{30} + \eta_{12}\right)^2 - \left(\eta_{21} + \eta_{03}\right)^2\right\}$$

Features

Internal Scalar Transform Shape Descriptors

Moment Invariants

The logarithm of ϕ_1 to ϕ_7 is normally used to reduce the dynamic range of the values when using these moment invariants as features in a classification task

Features

Internal Scalar Transform Shape Descriptors

- Shape descriptors based on moment invariants convey significant information for simple objects but fail to do so for complicated ones
- Since we are discussing internal scalar transform descriptors, it would seem that these moment invariants can only be generated from the entire region
- However, they can also be generated from the boundary contour of the object by exploiting Stokes' theorem or Green's theorem, both of which relate the integral over an area to an integral around its boundary (more later when discussing boundary chain codes BCC)

Features

External Space Domain Shape Descriptors

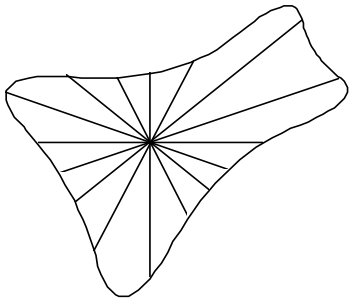
- Spatial structure of the boundary
- Example: syntactic descriptors of boundary primitives
    - short curves
    - line segments
    - corners
  - Shape descriptor is a string of primitive shapes
  - The string is constrained by the shape syntax or grammar
  - Shapes are recognized by parsing the string of primitive patterns

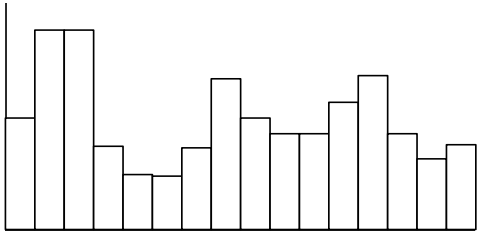
Features

External Space Domain Shape Descriptors

Polar radii signature

Recognition based on matching template signatures





Features

External Space Domain Shape Descriptors

Boundary Chain Code (BCC)

- More useful for shape representation rather than shape recognition
- The BCC encodes piecewise linear curves as a sequence of straight-line segments called Links

  A link a_i is a directed straight line segment of

  length $T\left(\sqrt{2}\right)^P$

  angle $a_i * 45°$ (referenced to the X-axis of a Cartesian co-ordinate system)
- T is the grid spacing and is normally set equal to unity

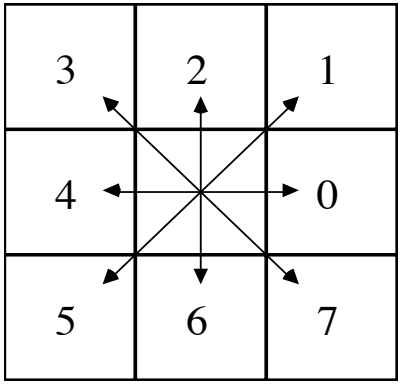
Features

External Space Domain Shape Descriptors

Boundary Chain Code (BCC)

- a_i is an integer in the range 0 to 7 and represents the (coarsely quantised) direction of the link
- p is the modulo two value of a_i; *i.e.*, $p = 0$ if a_i is even and $p = 1$ if a_i is odd

  Thus, the link length in directions 1, 3, 5 and 7 is equal to $\sqrt{2}$ and is equal to 1 in directions 0, 2, 4 and 6



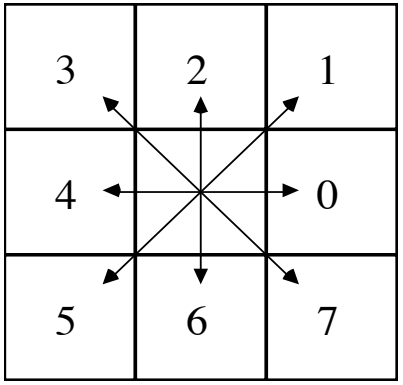
Features

External Space Domain Shape Descriptors

Boundary Chain Code (BCC)

A BCC is dependent on the orientation of the object boundary for two reasons:

1. Each link encodes the absolute direction of the boundary at that point
2. The link length (1 or $\sqrt{2}$) varies with the boundary direction



Features

External Space Domain Shape Descriptors

Boundary Chain Code (BCC)

The moment shape descriptors can be directly generated from the BCC

$$m_{00} = \frac{1}{2}\sum_{i=1} A_i$$

$$m_{10} = \frac{1}{3}\sum_{i=1}^{n} A_i\left(y_i - \frac{1}{2}\Delta y_i\right)$$

$$m_{01} = \frac{1}{3}\sum_{i=1}^{n} A_i\left(x_i - \frac{1}{2}\Delta x_i\right)$$

$$m_{11} = \frac{1}{4}\sum_{i=1}^{n} A_i\left(x_i y_i - \frac{1}{2}x_i\Delta y_i - \frac{1}{2}y_i\Delta x_i + \frac{1}{3}\Delta x_i\Delta y_i\right)$$

$$m_{02} = \frac{1}{4}\sum_{i=1}^{n} A_i\left(y_i^{\,2} - y_i\Delta y_i + \frac{1}{3}\Delta y_i^{\,2}\right)$$

$$m_{20} = \frac{1}{4}\sum_{i=1}^{n} A_i\left(x_i^{\,2} - x_i\Delta x_i + \frac{1}{3}\Delta x_i^{\,2}\right)$$

Features

External Space Domain Shape Descriptors

Boundary Chain Code (BCC)

x_{i-1} and y_{i-1} are the co-ordinates of a point on the perimeter of the shape

x_i and y_i are the co-ordinates of the subsequent point on the perimeter, as given by the BCC

Δx_i is defined to be $(x_i - x_{i-1})$

Δy_i is defined to be $(y_i - y_{i-1})$

A_i is defined to be $(x_i \Delta y_i - y_i \Delta x_i)$

n is the number of points on the boundary (*i.e.* the number of BCC links)

Features

Internal Space Domain Shape Descriptors

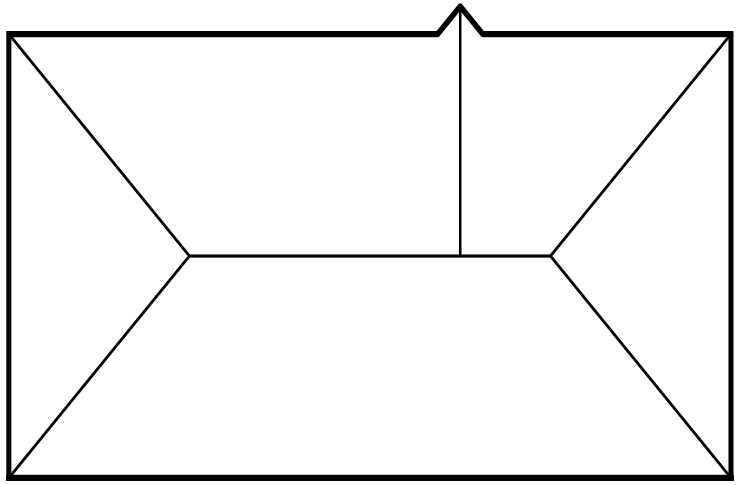
- Internal space domain techniques comprise descriptors which utilize structural or relational properties derived from the complete shape

Features

Internal Space Domain Shape Descriptors

The Medial Axis Transform (MAT)

- a skeleton representation of a two-dimensional shape
- A point in the shape is on the medial axis if and only if it is the centre of a circle which is a tangent to the shape boundary at two non-adjacent points
- Each point on the medial axis has a value associated with it which indicates the radius of this circle
- This represents the minimum distance to the boundary from that point and thus facilitates the reconstruction of the object



Features

Internal Space Domain Shape Descriptors

The Medial Axis Transform (MAT)

- Various methods for generating the medial axis
- Most intuitive of which is one that is often referred to as the Prairie Fire Technique
  - setting fire to the boundary of a dry grassy field
  - letting the flame burn inwards
  - The points at which the flame fronts meet are on the medial axis
- Sensitive to local distortions of the contour
- Small deviations can give to extraneous skeletal lines

Features

Internal Space Domain Shape Descriptors

Other descriptors can be derived using integral geometry:

- an object shape can be intersected by a number of chords in different directions
- the locations and the length of the intersection can be used in various ways as a shape descriptor

Features

Internal Space Domain Shape Descriptors

- Example: Normal Contour Distance (NCD) shape descriptor
  - A one-dimensional signature
  - Each signature value represents an estimate of the distance from a point on an object's boundary, with a local orientation of m, to the opposing boundary point which lies on a path whose direction is normal to m
  - The NCD signature is evaluated over the length of the available boundary
  - Does not require knowledge of the complete boundary and can be used for recognition of partially-occluded objects
  - If the contour represents a partial boundary, there may not be another boundary point which lies on a path which is normal to the contour and, hence, some segments of the NCD may be undefined

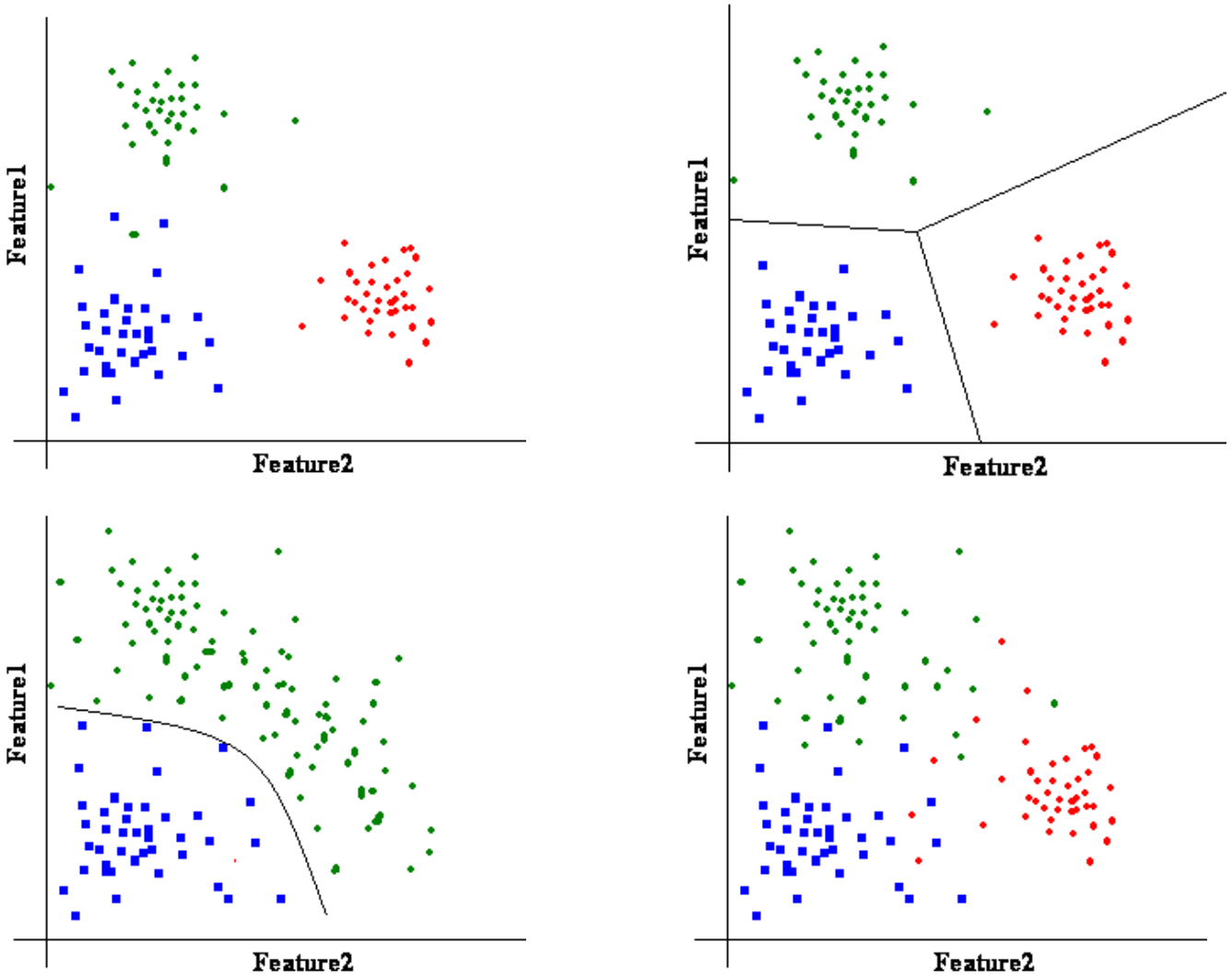
Classification

- The final stage of the statistical pattern recognition exercise
  - Classification of the objects on the basis of the set of features we have just computed, *i.e.* on the basis of the feature vector
  - Feature values are viewed as 'co-ordinates' of a point in n-dimensional space
  - Goal of classification: determine the sub-space to which the feature vector belongs
- Since each sub-space corresponds to a distinct object, the classification essentially accomplishes the object identification

Classification

- Object recognition
  - R classes: $w_1, w_2, \ldots w_R$
- Classifier
  - n features: input pattern / feature vector: $x_1, x_2, \ldots x_n$
- Feature space
  - Choosing the features
  - Clusters in feature space
- Separability
  - Linear separability
  - Hyper-surfaces
  - Inseparable classes
- Classifiers
  - Nearest Neighbour Classifier / Minimum Distance Classifier
  - Linear Classifier
  - (Naive) Maximum Likelihood Classifier / Naive Bayes Classifier
  - ...

Classification

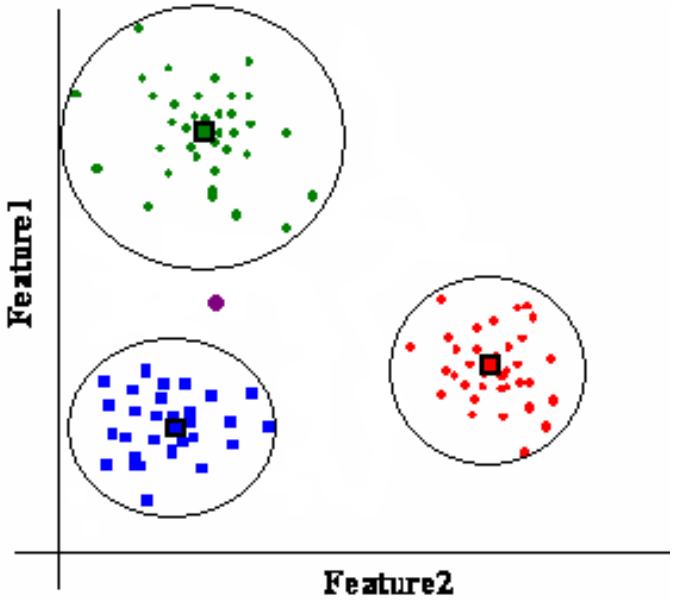


Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Classification

Nearest neighbour / minimum distance classifier

- Each class represented by an exemplar
- For an unknown object
  - Determine the distance to the exemplars
  - Pick the class with the smallest distance
- Unknown class?
  - Distance must be less than some threshold
- Advantages
  - Training
  - Computational complexity

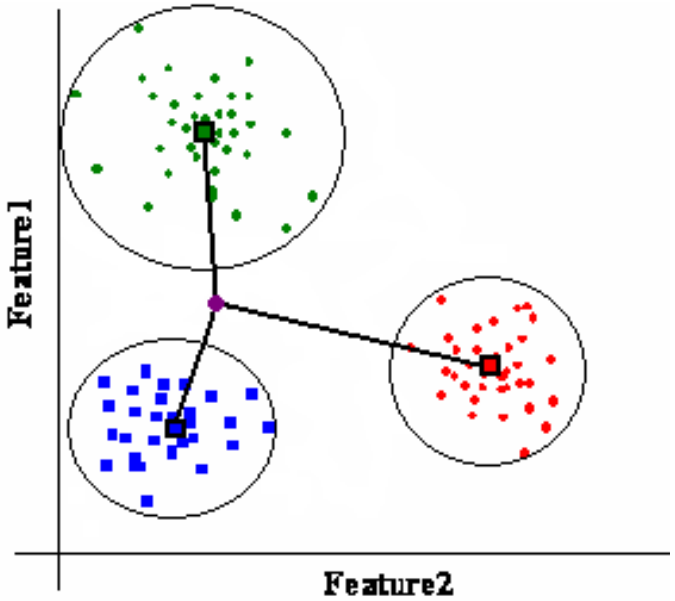


Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Classification

Nearest neighbour / minimum distance classifier

- Each class represented by an exemplar
- For an unknown object
  - Determine the distance to the exemplars
  - Pick the class with the smallest distance
- Unknown class?
  - Distance must be less than some threshold
- Advantages
  - Training
  - Computational complexity

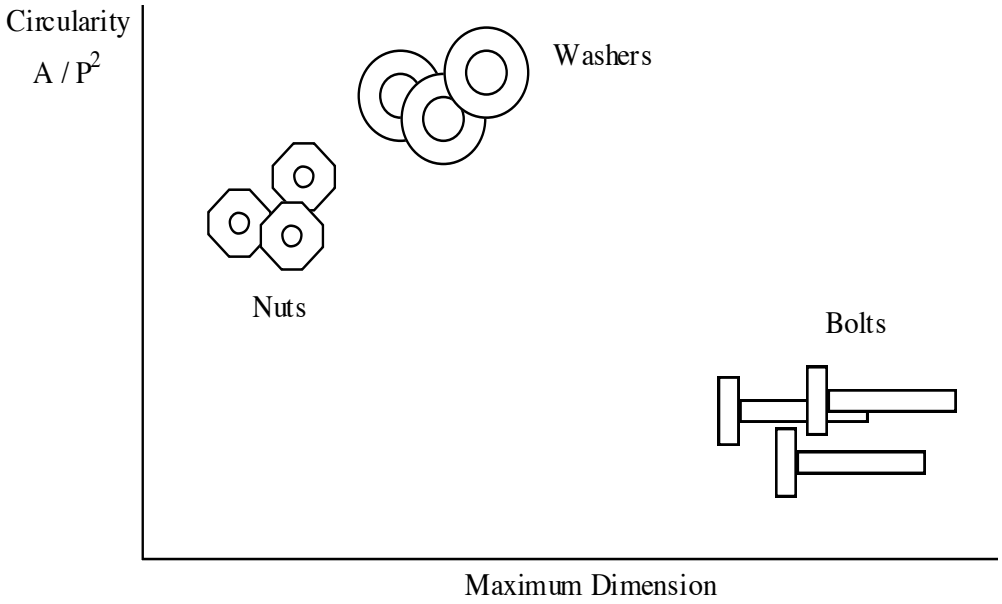


Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Classification

Example

- Classify nuts, bolts, and washers
- Use two features: circularity and maximum dimension



Classification

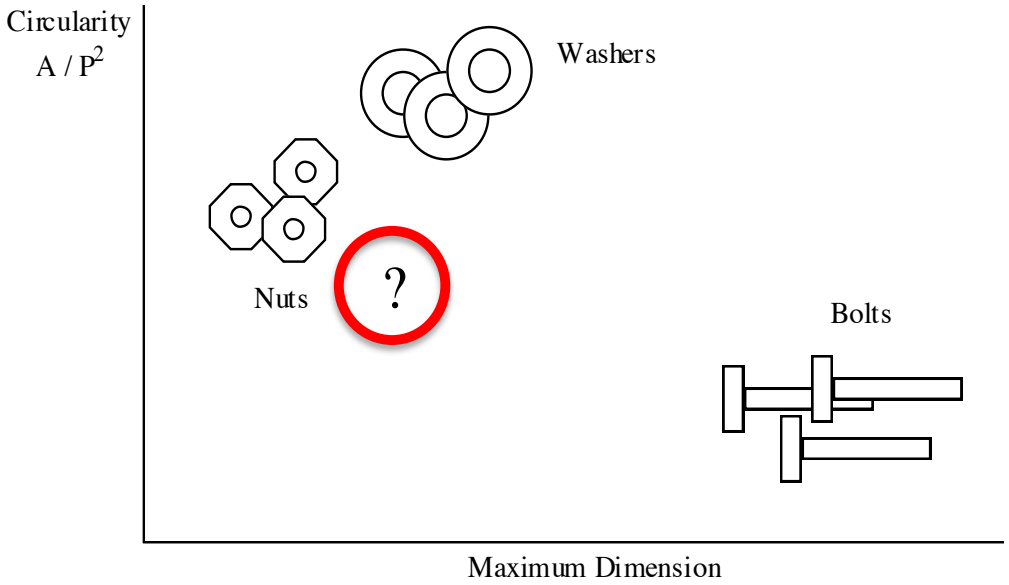
Example

- Classify nuts, bolts, and washers
- Use two features: circularity and maximum dimension
- Learn the distribution of these features for the three objects
  - Need a training set
- Measure the features for an unknown object
- Classify it on the basis of its position in the feature space
- Which sub-space does it belong to?

Classification

Example

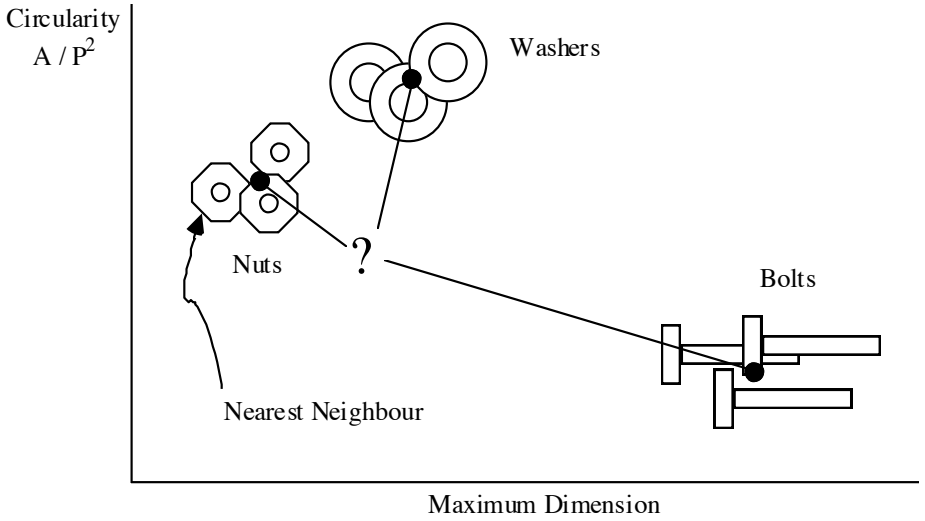
- Which sub-space does it belong to?



Classification

Nearest Neighbour Classification

- Classifies the object on the basis of the distance of the unknown object vector position from the centre of the three clusters
- Choosing the closest cluster as the one to which it belongs
- The position of the centre of each cluster is simply the average of each of the individual training vector positions



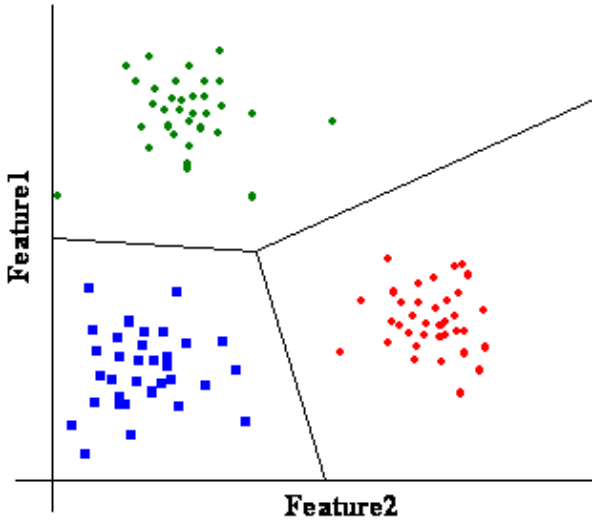
Classification

- Decision rule
  - $w_r = d(x)$ divides the feature space into R disjoint sub-spaces K_r, $r = 1, \ldots, R$

- Linear discrimination functions
  - For each class we can define a discrimination function $g_r(x)$ for which $g_r(x) \geq g_s(x)$ for all values of x for any point in K_r

    $$g_r(x) = q_{ro} + q_{r1}x_1 + \ldots + q_{rn}x_n$$

  - The hyper-surface between any two sub-spaces is defined as $g_r(x) - g_s(x) = 0$

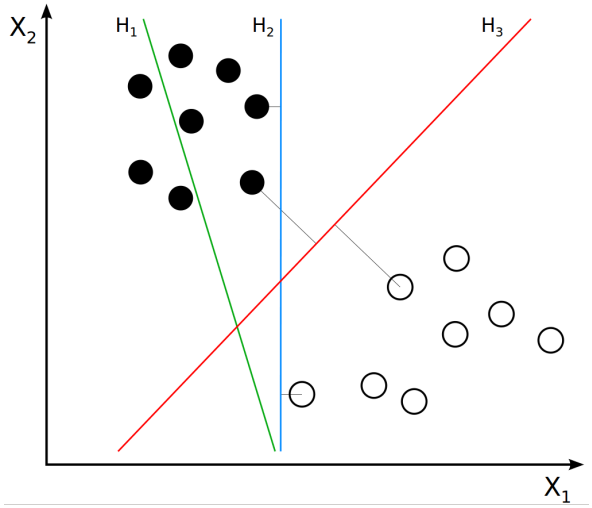


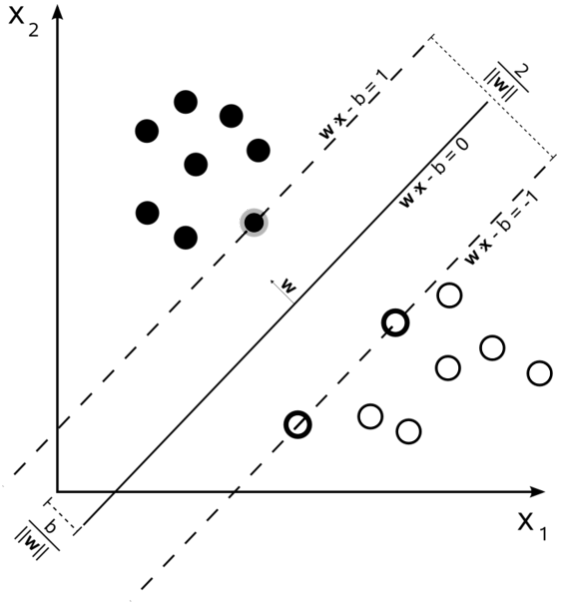
- Unknown class
  - The discrimination function which has the highest value defines which class is selected: $g_r(x) > \text{threshold}$

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Classification

Support Vector Machine (SVM) classification





- Maximum margin classifier: H3 separates classes “better” than H2
- Training examples defining separating hyperplane: support vectors
- If not linearly separable: use kernel trick to map to higher-dimensional space

Credit: Markus Vincze, Technische Universität Wien

Classification

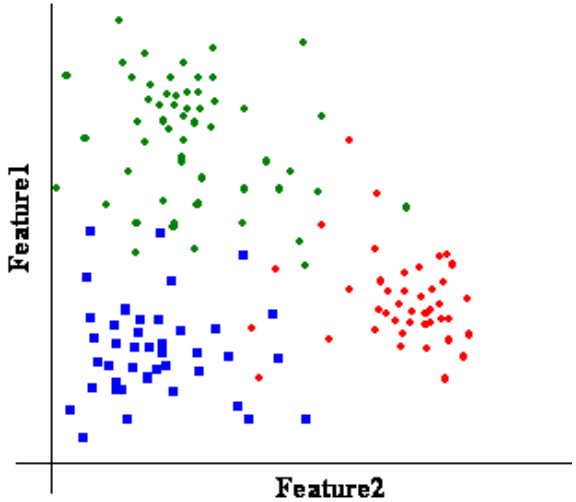
Classifier Learning

- Training set
  - Must be representative
  - Must be inductive
- Training set size
  - Training set provides the unknown statistical information
  - Size will typically have to be increased several times
- Sample learning strategies
  - Supervised:
    - Probability density estimation – estimating $p(x \mid w_r)$ & $p(w_r)$
    - Training set includes class specification for every instance
  - Unsupervised:
    - Cluster Analysis
    - Look for similarities in feature space

Classification

Maximum-likelihood classifier

- Use knowledge about the statistical distribution of each class
- Classification maximizes the probability of assigning the unknown object to the correct class



Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Classification

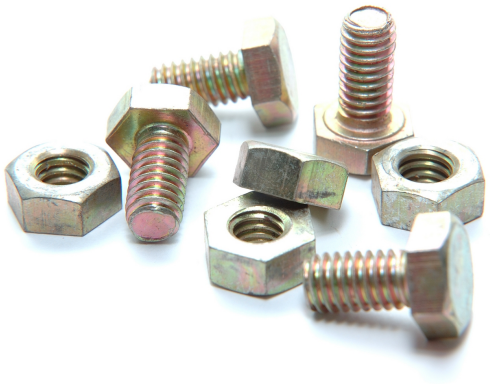
Maximum-likelihood classifier

- Suppose we wish to distinguish between nuts and bolts (no washers this time).
- Circularity measure will suffice
  - one feature and a one-dimensional feature space
  - two classes of object: nuts and bolts.
- Let us refer to these classes as C_n and C_b
- Let us refer to the circularity feature value as x
- We can use Bayes' Theorem to create a good classifier (better than a nearest neighbour classifier)

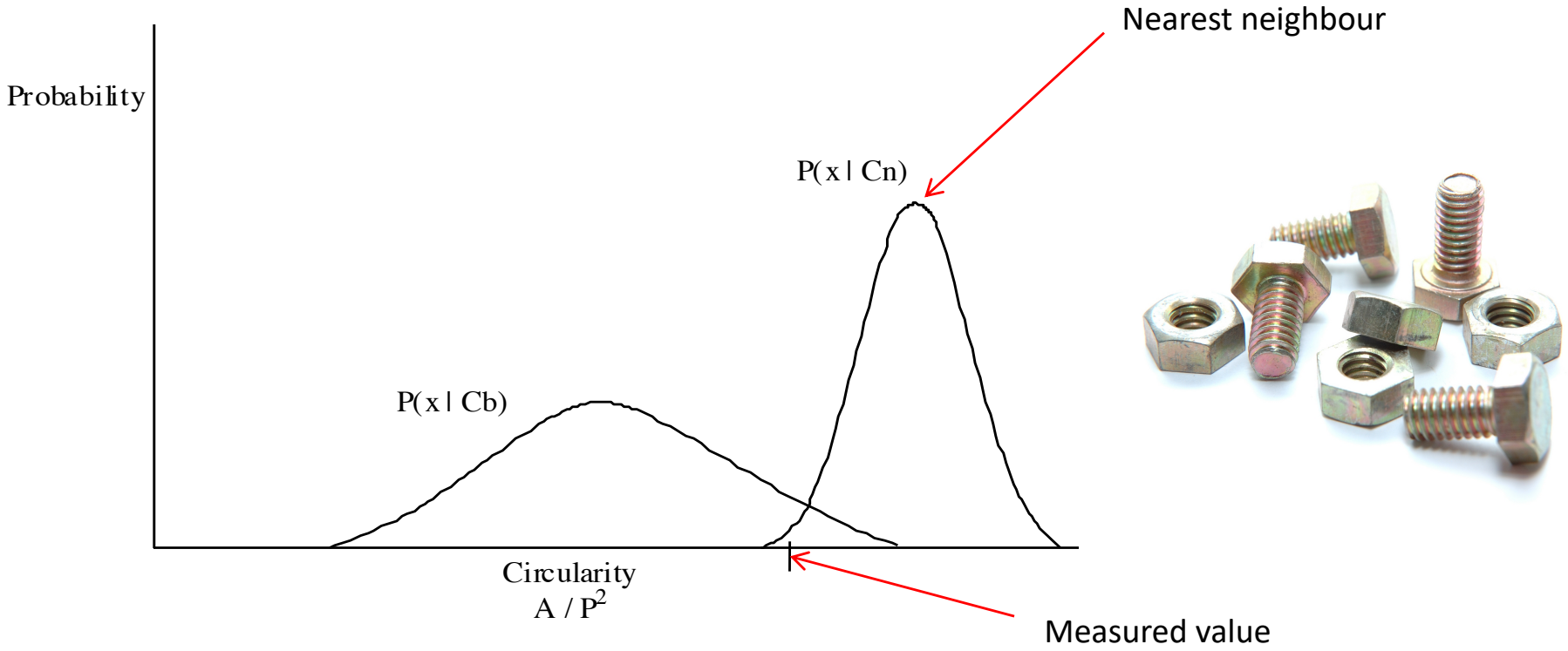
Classification

Example: Maximum Likelihood Classifier

- Let's design a system that can classify two parts: nuts and bolts
- Two classes $C_b\, C_n$
- Let's decide to use a feature 'circularity' x to distinguish nuts from bolts (nuts are more circular than bolts)

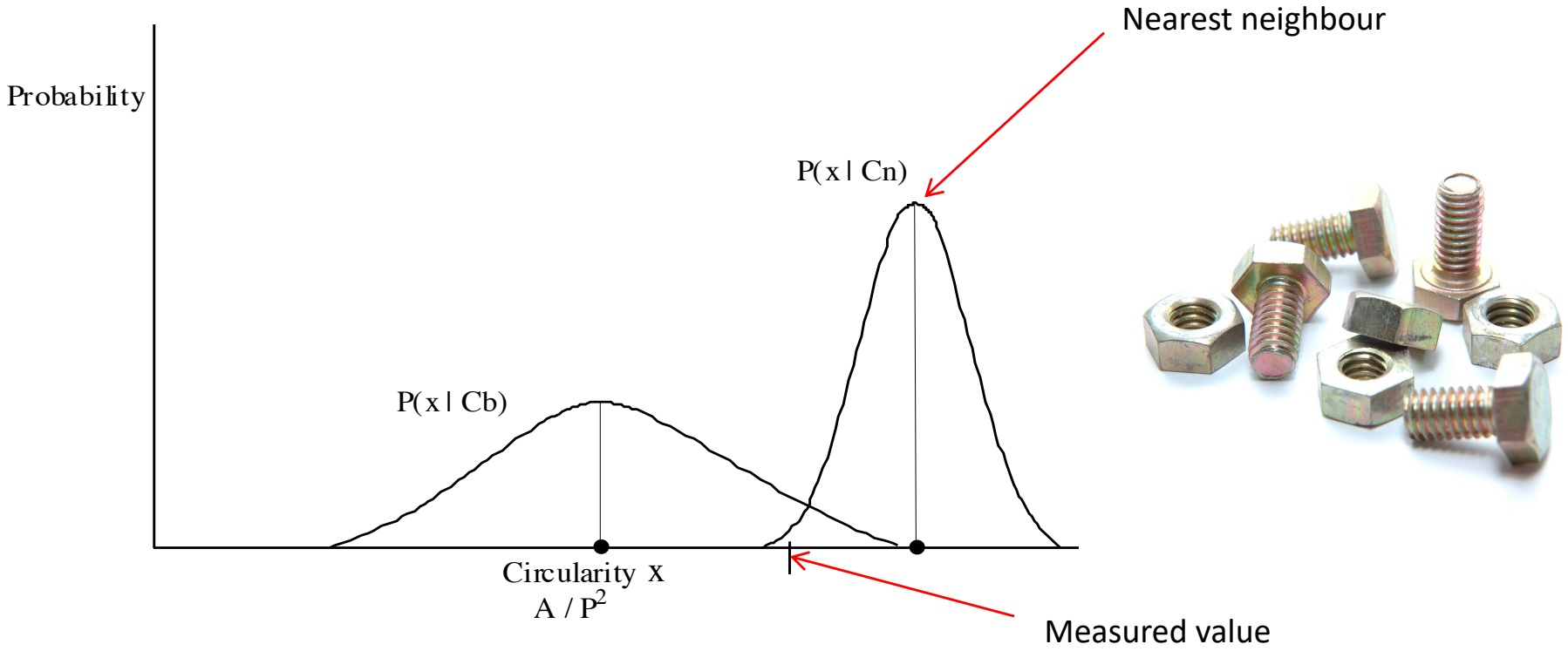


Classification



We can do better than nearest neighbour if we use knowledge about the probability of an object having some feature value $P(x|C_i)$ and the probability of that object being there at all $P(C_i)$

Classification



We can do better than nearest neighbour if we use knowledge about

the probability of an object having some feature value $P(x|C_i)$

and the probability of that object being there at all $P(C_i)$

Classification

Example: Maximum Likelihood Classifier

- Neither of these probabilities are what we are interested in!
- We want the probability that an object belongs to a particular class, *given that a particular value of x has occurred* $P(C_i|x)$
- Thus, we classify the object as a bolt if

$$P(C_b|x) > P(C_n|x)$$

- We use Bayes' Theorem to convert the probabilities we know (or can measure) to the ones we need

Classification

Example: Maximum Likelihood Classifier

- The *posterior* probability, $P(C_i|x)$, that the object belongs to a particular class i is given by Bayes' Theorem:

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

where

$$P(x) = \sum_{i=1}^{2} P(x|C_i)P(C_i)$$

Classification

Example: Maximum Likelihood Classifier

- The first thing that is required is the probabilities for each of these two classes, *i.e.*, a measure of the probabilities that an object from a particular class will have a given feature value

- Since it is not likely that we will know these *a priori*, we will have to estimate them

Classification

Example: Maximum Likelihood Classifier

- Let S be the space of circularity values that we can measure with our computer vision system
- Let $X_n(x)$ be the random variable that equals the number of times a given circularity value appears when x is the outcome (i.e. when the circularity of a nut is measured)
- Let $X_b(x)$ be the random variable that equals the number of times a given circularity value appears when x is the outcome (i.e. when the circularity of a bolt is measured)
- We need the probability distribution of random variables X_n and X_b

Classification

Example: Maximum Likelihood Classifier

- We have used discrete random variables here
  - The distribution is called a **Probability Mass Function**
- However, since the circularity value is going to vary continuously (i.e. it won't have a finite set of values), we should really use a continuous random variable
  - The distribution is call a **Probability Density Function (PDF)**

Classification

Example: Maximum Likelihood Classifier

- The PDF for nuts can be estimated in a relatively simple manner
  - measuring the value of x for a large number of nuts
  - plotting the histogram of these values
  - smoothing the histogram
  - normalising the values so that the total area under the histogram equals 1
    - The normalisation step is necessary because certainty has a probability value of 1 and the sum of all the probabilities (for all the possible circularity measures) must necessarily be equal to a certainty of having that object, *i.e.*, a probability value of 1

Classification

Example: Maximum Likelihood Classifier

- The PDF for the bolts can be estimated in a similar manner.
- Next problem: the probability of each class occurring

  - We may know, for instance, that the class of nuts is, in general, likely to occur twice as often as the class of bolts
  - In this case we say that the prior (or *a priori*) probability of the two classes are :

    $P(C_n) = 0.666$ and $P(C_b) = 0.333$

  - In fact, in this case, it is more likely that they will have the same *a priori* probabilities (0.5) since we usually have a nut for each bolt

Classification

Example: Maximum Likelihood Classifier

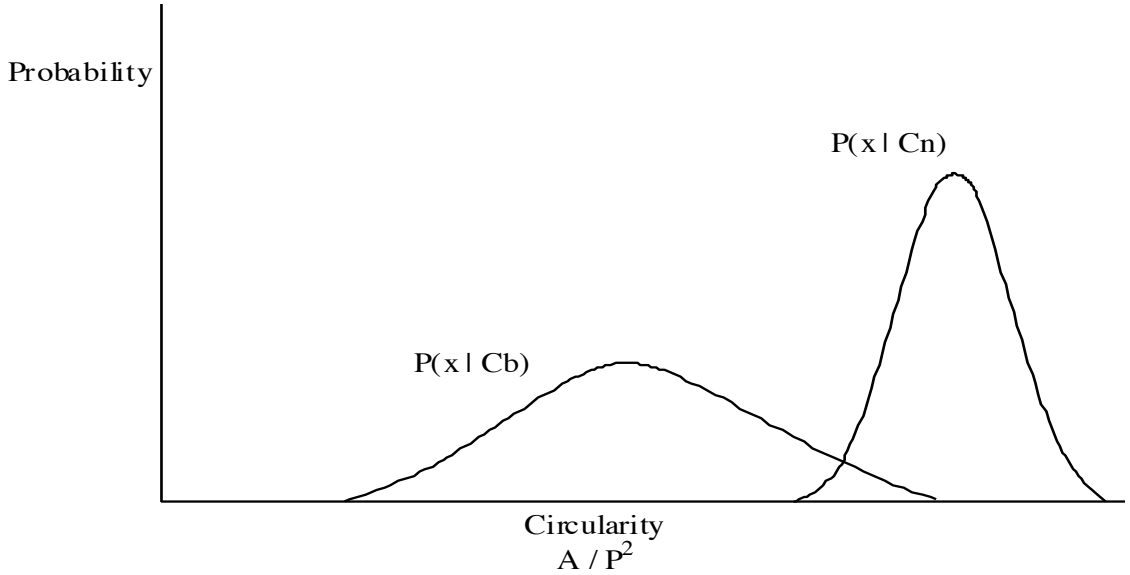
- The PDFs tell us the probability that the circularity x will occur, given that the object belongs to the class of nuts C_n in the first instance and to the class of bolts C_b in the second instance

- As we know, this is termed the conditional probability of an object having a certain feature value, given that we know that it belongs to a particular class

Classification

Example: Maximum Likelihood Classifier

- Thus, the conditional probability, $p(x \mid C_b)$ enumerates the probability that a circularity x will occur, given that the object is a bolt.
- The two conditional probabilities $p(x \mid C_b)$ and $p(x \mid C_n)$ are shown below



Classification

Example: Maximum Likelihood Classifier

- This is not what are interested in …
- We want the probability that an object belongs to a particular class, given that a particular value of x has occurred (*i.e.* been measured), allowing us to establish its identity

Classification

Example: Maximum Likelihood Classifier

- This is called the posterior (or *a posteriori*) probability, $p(C_i|x)$ that the object belongs to a particular class C_i, given that a particular value of x has occurred
- It is given by Bayes' Theorem

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

where

$$P(x) = \sum_{i=1}^{2} P(x|C_i)P(C_i)$$

Classification

Example: Maximum Likelihood Classifier

- $p(x)$ is a normalisation factor which is used to ensure that the sum of the *a posteriori* probabilities sum to one, for the same reasons as mentioned earlier

Classification

Example: Maximum Likelihood Classifier

- In effect, Bayes' theorem allows us to use
  - the *a priori* probability of objects occurring in the first place
  - the conditional probability of an object having a particular feature value given that it belongs to a particular class and ...
  - The actual measurement of a feature value (to be used as the parameter in the conditional probability) to estimate the probability that the measured object belongs to a given class
  - Once we can estimate the probability that, for a given measurement, the object is a nut and the probability that it is a bolt, we can make a decision as to its identity, choosing the class with the higher probability

Classification

Example: Maximum Likelihood Classifier

- This is why it is called the maximum likelihood classifier
- Thus, we classify the object as a bolt if :

$$P\left(C_b|x\right) > P\left(C_n|x\right)$$

Classification

Example: Maximum Likelihood Classifier

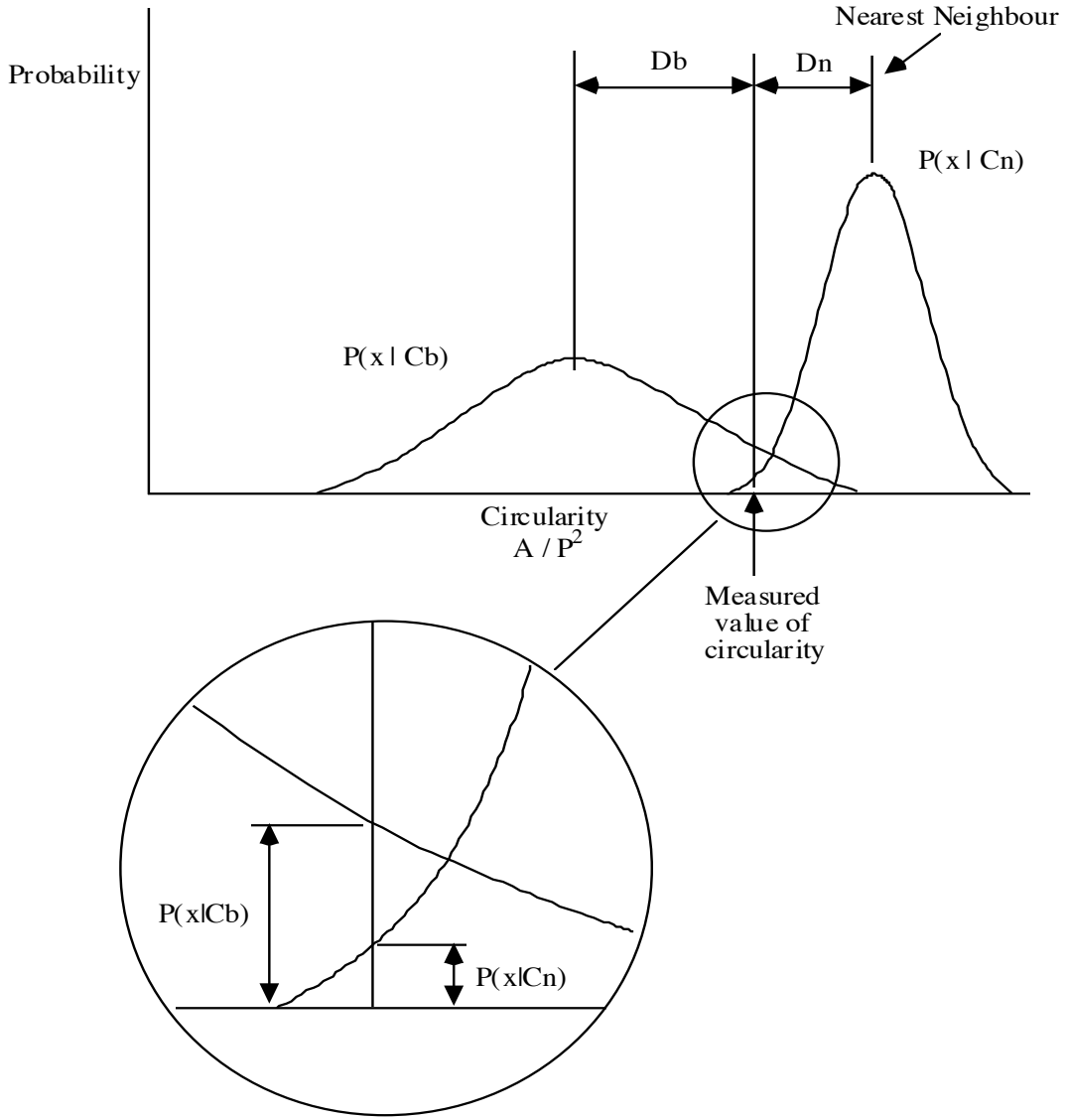
- Using Bayes' Theorem again, and noting that the normalising factor $p(x)$ is the same for both expressions, we can rewrite this test as

$$P(x|C_b)P(C_b) > P(x|C_n)P(C_n)$$

- If we assume that the chances of an unknown object being either a nut or a bolt are equally likely [*i.e.* $P(C_b) = P(C_n)$], then we classify the unknown object as a bolt if :

$$P(x|C_b) > P(x|C_n)$$

Classification



Classification

Example: Maximum Likelihood Classifier

- For the example shown $p(x \mid C_b)$ is indeed greater than $p(x \mid C_n)$ for the measured value of circularity and we classify the object as a bolt

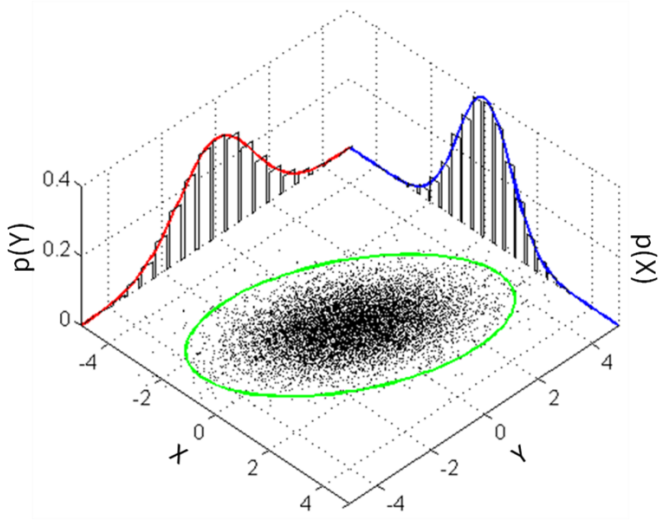
- If, on the other hand, we were to use the nearest neighbour classification technique, we would choose the class whose mean value "is closer to" the measured value

  - In this case, the distance D_n from the measured value to the mean of the PDF for nuts is less than D_b, the distance from the measured value to the mean of the PDF for bolts; we would erroneously classify the object as a nut

Classification

Example: Maximum Likelihood Classifier

- This was a simple example with just one feature and a 1-D PDF
- However, the argument generalizes directly to n-dimensions, where we have n features in which case the conditional probability density functions are also n-dimensional



Classification

Example: Maximum Likelihood Classifier

- If we assume that the features are independent then we can use the theory we've just outlined, multiplying together the conditional probabilities for each class
  - This is known a **Naïve Bayes Classifier**
  - It may be naïve, but it works surprisingly well
- If we don't assume independence, then we need a more complex theory

Demos

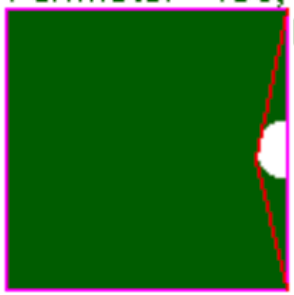
The following code is taken from the featureExtraction project in the lectures directory of the ACV repository

See:

```
featureExtraction.h
featureExtractionImplementation.cpp
featureExtractionApplication.cpp
```

Perimeter=396, Area=10000, BArea=10000, CArea=10000
HuMoments = 0.17, 0.00, 0.00

Perimeter=406, Area=9814, BArea=10005, CArea=10005
HuMoments = 0.17, 0.00, 0.00



Perimeter=284, Area=8000, BArea=9944, CArea=8060
HuMoments = 0.16, 0.00, 0.00

Perimeter=291, Area=7837, BArea=9947, CArea=8055
HuMoments = 0.16, 0.00, 0.00

```
/*
  Example use of openCV to perform 2D feature extraction
  ------------------------------------------------------
  Implementation file

  David Vernon
  18 June 2017
*/

#include "featureExtraction.h"

void featureExtraction(char *filename, FILE *fp_out) {

   char inputWindowName[MAX_STRING_LENGTH]              = "Input Image";
   char outputWindowName[MAX_STRING_LENGTH]             = "Contour Image";

   Mat inputImage;

   namedWindow(inputWindowName,   CV_WINDOW_AUTOSIZE);
   namedWindow(outputWindowName,  CV_WINDOW_AUTOSIZE);

   inputImage = imread(filename, CV_LOAD_IMAGE_COLOR); // Read the file

   if (!inputImage.data) {                           // Check for invalid input
      printf("Error: failed to read image %s\n",filename);
      prompt_and_exit(-1);
   }

   printf("Press any key to continue ...\n");

   fprintf(fp_out,"%s \n",filename);  // file write added by David Vernon
```

```
/*
 * The following is based on code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
 * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
 */

  /* convert the input image to a binary image */
  Mat gray;
  Mat binary;

  cvtColor(inputImage, gray, CV_BGR2GRAY);
  //threshold(gray,binary,128,255,THRESH_BINARY_INV);
  threshold(gray,binary,128, 255,THRESH_BINARY_INV  | THRESH_OTSU); // David Vernon: substituted in automatic threshold selection

  /* extract the contours of the objects in the binary image */
  vector<vector<Point>> contours;
  vector<Vec4i>         hierarchy;

  /* David Vernon: see http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#findcontours */
  findContours(binary,contours,hierarchy,CV_RETR_TREE,CV_CHAIN_APPROX_NONE);

  /* extract features from the contours */
  Mat contours_image = Mat::zeros(binary.size(), CV_8UC3);
  contours_image = Scalar(255,255,255);

  //binary.copyTo(contours_image,binary);

  /* Prepare to do some processing on all contours (objects and holes!) by declaring appropriate data-structures */
  vector<RotatedRect>   min_bounding_rectangle(contours.size());   // bounding rectangles
  vector<vector<Point>> hulls(contours.size());                    // convex hulls
  vector<vector<int>>   hull_indices(contours.size());             // indices of convex hulls
  vector<vector<Vec4i>> convexity_defects(contours.size());        // convex cavities
  vector<Moments>       contour_moments(contours.size());          // Hu moments
```

C++ ... in C it's a struct

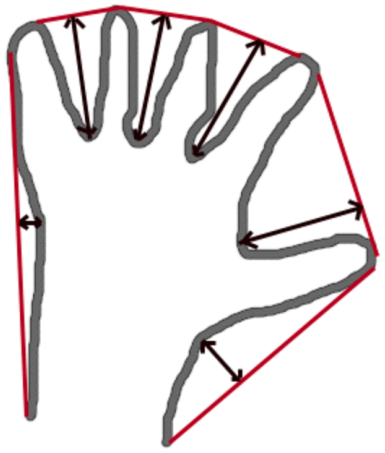
```
for (int contour_number=0; (contour_number<(int)contours.size()); contour_number++) {
   if (contours[contour_number].size() > 10) { // only consider contours of appreciable length

      /* David Vernon: see http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#boundingrec
      min_bounding_rectangle[contour_number] = minAreaRect(contours[contour_number]);

      /* David Vernon: see http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#convexhull
      convexHull(contours[contour_number], hulls[contour_number]);
      convexHull(contours[contour_number], hull_indices[contour_number]);

      /* David Vernon: see http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#convexityde
      convexityDefects(contours[contour_number], hull_indices[contour_number], convexity_defects[contour_number]);

      /* David Vernon: see http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#moments*/
      contour_moments[contour_number] = moments( contours[contour_number] );
   }
}
```

C:

```
struct CvConvexityDefect
{
   CvPoint* start; // point of the contour where the defect begins
   CvPoint* end; // point of the contour where the defect ends
   CvPoint* depth_point; // the farthest from the convex hull point within the defect
   float depth; // distance between the farthest point and the convex hull
};
```

Fixed-point approximation (with 8 fractional bits) of the distance between the farthest contour point and the hull. That is, the floating-point value of the depth depth/256.0

The figure below displays convexity defects of a hand contour:



http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#convexitydefects

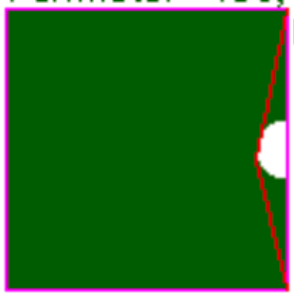
```
/* for all contours */
for (int contour_number=0; (contour_number>=0); contour_number=hierarchy[contour_number][0]) {

   /* only consider contours of appreciable length */
   if (contours[contour_number].size() > 10) {
      Scalar colour(rand()&0x7F, rand()&0x7F, rand()&0x7F );                                    // generate a random colour
      drawContours(contours_image, contours, contour_number, colour, CV_FILLED, 8, hierarchy ); // draw the contour

      char output[500];

      // David Vernon: Ken Dawson-Howe adjusts area as it seems to be underestimated by half the number of pixels on the perimeter
      double area = contourArea(contours[contour_number]) + contours[contour_number].size()/2 + 1;

      // Process any holes (removing the area from the area of the enclosing contour)
      for (int hole_number=hierarchy[contour_number][2]; (hole_number>=0); hole_number=hierarchy[hole_number][0]) {

         // David Vernon: Ken Dawson-Howe adjusts area as it seems to be underestimated by half the number of pixels on the perimeter
         area -= (contourArea(contours[hole_number]) - contours[hole_number].size()/2 + 1);

         Scalar colour( rand()&0x7F, rand()&0x7F, rand()&0x7F );
         drawContours( contours_image, contours, hole_number, colour, CV_FILLED, 8, hierarchy );

         sprintf(output,"Area=%.0f", contourArea(contours[hole_number]) -contours[hole_number].size()/2+1);

         /* write to file added by David Vernon */
         fprintf(fp_out,"Object %d, Hole %d: Area = %.0f\n",
                 contour_number, hole_number, contourArea(contours[hole_number]) - contours[hole_number].size()/2 + 1);

         Point location( contours[hole_number][0].x + 20, contours[hole_number][0].y + 5 );
         putText( contours_image, output, location, FONT_HERSHEY_SIMPLEX, 0.4, colour );
      }
```

```
/* Draw the minimum bounding rectangle */
Point2f bounding_rect_points[4];
min_bounding_rectangle[contour_number].points(bounding_rect_points);
line(contours_image, bounding_rect_points[0], bounding_rect_points[1], Scalar(0, 0, 127));
line(contours_image, bounding_rect_points[1], bounding_rect_points[2], Scalar(0, 0, 127));
line(contours_image, bounding_rect_points[2], bounding_rect_points[3], Scalar(0, 0, 127));
line(contours_image, bounding_rect_points[3], bounding_rect_points[0], Scalar(0, 0, 127));

float bounding_rectangle_area = min_bounding_rectangle[contour_number].size.area();

/* Draw the convex hull */
drawContours(contours_image, hulls, contour_number, Scalar(255,0,255) );  // purple

/* Highlight any convexities */
int largest_convexity_depth=0;

for (int convexity_index=0; convexity_index < (int)convexity_defects[contour_number].size(); convexity_index++) {
   if (convexity_defects[contour_number][convexity_index][3] > largest_convexity_depth)
      largest_convexity_depth = convexity_defects[contour_number][convexity_index][3];

   if (convexity_defects[contour_number][convexity_index][3] > 256*2) {
      line( contours_image, contours[contour_number][convexity_defects[contour_number][convexity_index][0]],
                            contours[contour_number][convexity_defects[contour_number][convexity_index][2]], Scalar(0,0, 255));
      line( contours_image, contours[contour_number][convexity_defects[contour_number][convexity_index][1]],
                            contours[contour_number][convexity_defects[contour_number][convexity_index][2]], Scalar(0,0, 255));
   }
}
```

```
        //sprintf(output,"Perimeter=%d, Area=%.0f, BArea=%.0f, CArea=%.0f", contours[contour_number].size(),area,min_bounding_rectangle[c
        /* David Vernon: alternative as area seems to be underestimated by half the number of pixels on the perimeter */
        sprintf(output,"Perimeter=%d, Area=%.0f, BArea=%.0f, CArea=%.0f", contours[contour_number].size(),
                                                                          area,
                                                                          min_bounding_rectangle[contour_number].size.area() + contours[c
                                                                          contourArea(hulls[contour_number])+ contours[contour_number].si
        /* file write added by David Vernon */
        /* David Vernon: area seems to be underestimated by half the number of pixels on the perimeter */
        fprintf(fp_out,"Object %d: perimeter = %d, object area = %.0f, bounding rectangle area = %.0f, convex hull area = %.0f \n",
                contour_number,
                contours[contour_number].size(),
                area,
                min_bounding_rectangle[contour_number].size.area() + contours[contour_number].size()/2 + 1,
                contourArea(hulls[contour_number]) + contours[contour_number].size()/2 + 1);

        Point location( contours[contour_number][0].x, contours[contour_number][0].y-3 );
        putText(contours_image, output, location, FONT_HERSHEY_SIMPLEX, 0.4, colour );

        /* David Vernon: see http://docs.opencv.org/2.4.10/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#humoments *
        double hu_moments[7];
        HuMoments(contour_moments[contour_number], hu_moments );

        sprintf(output,"HuMoments = %.2f, %.2f, %.2f", hu_moments[0],hu_moments[1],hu_moments[2]);
        Point location2( contours[contour_number][0].x+100, contours[contour_number][0].y-3+15 );
        putText(contours_image, output, location2, FONT_HERSHEY_SIMPLEX, 0.4, colour );

        /* filewrite added by David Vernon */
        fprintf(fp_out,"Object %d: HuMoments = %.2f, %.2f, %.2f \n\n", contour_number, hu_moments[0],hu_moments[1],hu_moments[2]);
      }
      fprintf(fp_out,"\n"); //file write added by David Vernon
   }
   imshow(inputWindowName,  inputImage );
   imshow(outputWindowName, contours_image);

   do{
      waitKey(30);
   } while (!_kbhit());

   getchar(); // flush the buffer from the keyboard hit

   destroyWindow(inputWindowName);
   destroyWindow(outputWindowName);
}
```

Perimeter=396, Area=10000, BArea=10000, CArea=10000
HuMoments = 0.17, 0.00, 0.00

Perimeter=406, Area=9814, BArea=10005, CArea=10005
HuMoments = 0.17, 0.00, 0.00



Perimeter=284, Area=8000, BArea=9944, CArea=8060
HuMoments = 0.16, 0.00, 0.00

Perimeter=291, Area=7837, BArea=9947, CArea=8055
HuMoments = 0.16, 0.00, 0.00


0
Perimeter=80, Area=409, BArea=692, CArea=596
Area=172
HuMoments = 0.17, 0.00, 0.00
1
Perimeter=101, Area=275, BArea=573, CArea=480
HuMoments = 0.46, 0.13, 0.01
2
Perimeter=111, Area=347, BArea=626, CArea=584
HuMoments = 0.42, 0.07, 0.00
3
Perimeter=121, Area=353, BArea=650, CArea=592
HuMoments = 0.40, 0.06, 0.00
4
Perimeter=86, Area=350, BArea=636, CArea=488
Area=59
HuMoments = 0.20, 0.01, 0.00
5
Perimeter=121, Area=333, BArea=631, CArea=565
HuMoments = 0.42, 0.06, 0.00
6
Perimeter=101, Area=395, BArea=671, CArea=587
HuMoments = 0.23, 0.01, 0.00
Area=77
7
Perimeter=94, Area=269, BArea=599, CArea=442
HuMoments = 0.46, 0.09, 0.04
8
Perimeter=85, Area=443, BArea=694, CArea=615
Area=45
Area=59
HuMoments = 0.20, 0.01, 0.00
9
Perimeter=101, Area=394, BArea=640, CArea=575
Area=70
HuMoments = 0.23, 0.01, 0.00

Reading

D. Vernon, *Machine Vision: Automated Visual Inspection and Robot Vision*, Prentice-Hall, 1991.

Section 6.3 Decision-theoretic Techniques

Aside: Probability

Probability provides a mathematical foundation for many concepts

- Information
- Belief
- Uncertainty
- Confidence
- Randomness
- Variability
- Chance
- Risk

Aside: Probability

Probability Theory provides

- A framework for making inferences and testing hypotheses based on uncertain empirical data
- Building systems that operate in an uncertain world
  - Machine perception (speech recognition, computer vision)
  - Artificial intelligence
- Theoretical framework for understanding how the brain works
  - Many computational neuroscientists think the brain is a probabilistic computer build with unreliable components (i.e. neurons)

Aside: Probability

Probability Theory provides

- **A way of combining different sources of uncertain information to make rational decisions**

Aside: Probability

Three major interpretations of probability

Frequentist: probability as a relative frequency

- Probability of an event as the proportion of times such an event is expected to happen in the long run.

- The probability of an event E would be the limit of the relative frequency of occurrence of that event as the number of observations grows large

Number of times the event is observed

$$P(\mathrm{E}) = \lim_{n \to \infty} \frac{n_{\mathrm{E}}}{n}$$

Number of independent experiments

Aside: Probability

Three major interpretations of probability

Frequentist: probability as a relative frequency

- Appealing: objective, ties in with work on observation of physical events
- Can't perform an experiment an infinite number of times
- Behaviourist approach: based on observable behaviour of physical systems
- Doesn't capture idea of probability as internal knowledge of cognitive systems

Aside: Probability

Three major interpretations of probability

Bayesian or subjectivist: probability as uncertain knowledge

- "I will probably get an A in this class"

  By which we mean, "based on what I know about myself and about this class, I would not be very surprised if I get an A. However, I wouldn't bet my life on it, since there are a multitude of factors which are difficult to predict and that could make it impossible for me to get an A"

- This notion of probability is cognitive and does not need to be grounded in empirical frequencies

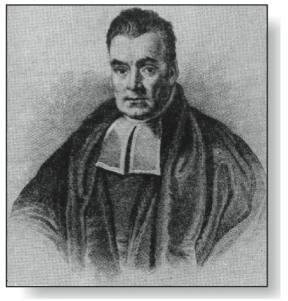
  "I will probably die poor" … not able to repeat that experiment many times and count the number of lives in which I die poor

Aside: Probability

Three major interpretations of probability

Bayesian or subjectivist:

- Useful in the field of machine intelligence
- Need to have knowledge systems capable of handling the uncertainty of the world
- Probabilists that are willing to represent internal knowledge using probability theory are called "Bayesian" (since Bayes was the first mathematician to do so)



THOMAS BAYES (1702–1761)

Aside: Probability

Three major interpretations of probability

Axiomatic or mathematical: probability as a mathematical model

- Rigorous definition
- Traceable to first principles
- Avoid the frequentist vs. Bayesian debate
- Application of probability theory is not the main concern

Aside: Probability

Experiments with finitely many, equally likely, outcomes

- **Experiment**: a procedure that yields one of a given set of possible outcomes
  - E.g. rolling a die, tossing a coin, tossing a coin two times
- **Sample space** S: set of possible outcomes
  - E.g. Ω = {1,2,3,4,5,6}, Ω = {H, T}, Ω = {{H,H}, {H, T}, {T, H}, {T, T}}
  - Also called **outcome space** or **reference set**
- **Event** E: subset of the sample space
  - Sets of outcomes
  - The set of all events is called the **event space**
  - E.g. Rolling an even number on a die: E = {2, 4, 6}

Aside: Probability

Definition of probability

For an event E, and a sample space S

The probability of E is $$p(E) = \frac{|E|}{|S|}$$

The probability of an event is between 0 and 1

– Example: an box contains four blue balls and five red balls; what is the probability that a ball chosen at random for the box is blue?

9 possible outcomes, four produce a blue ball, so probability is $4/9$

Aside: Probability

Probabilities of Complements and Unions of Events

For an event E, and a sample space S

The probability of the complement of E, $\overline{E} = S - E$, is given by

$$p(\overline{E}) = 1 - p(E)$$

– Example: A sequence of 10 bits is randomly generated. What is the probability that at least one of these bits is 0?

Let E be the event with at least one of the 10 bits is 0

Then $\overline{E}$ is the event that all the bits are 1.

The sample space S is the set of all strings of length 10,

Aside: Probability

Probabilities of Complements and Unions of Events

For an event E, and a sample space S

The probability of the complement of E, $\overline{E} = S - E$, is given by

$$p(\overline{E}) = 1 - p(E)$$

– Example: A sequence of 10 bits is randomly generated. What is the probability that at least one of these bits is 0?

$$p(E) = 1 - p(\overline{E}) = 1 - \frac{|\overline{E}|}{|S|} = 1 - \frac{1}{2^{10}}$$

$$= 1 - \frac{1}{1024} = \frac{1023}{1024}.$$

Aside: Probability

Probability measures

- You can think of probability as a function that assigns a number to a set: probability 'measures' a set (hence probability measures)

- If events $E_1, E_2, \ldots \; E_n$ are disjoint (i.e. no elements in common)

$$p(E_1 \cup E_2 \ldots \cup E_n) = p(E_1) + \; p(E_2) + \; \ldots \; p(E_n)$$

- Probability of rolling a die and getting a 1: $p\,(\{1\}) = 1/6$
  same for 2, 3, 4, 5, and 6.

$$p(\{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\} \cup \{6\}) = 1/6 + 1/6 + 1/6 + 1/6 + 1/6 + 1/6 = 1$$

Aside: Probability

Probabilities of Intersection of Events: Joint Probability

- For events E_1 and E_2 in a sample space S

$$p(E_1, E_2) = p(E_1 \cap E_2)$$

- The joint probability of two or more events is the probability of the intersection of those events

Aside: Probability

Probabilities of Intersection of Events: Joint Probability

- Consider the event E_1 = {2, 4, 6} when rolling a die (rolling an even number)
- Consider the event E_2 = {4, 5, 6} (rolling a number greater than 3)
- The joint probability (rolling an even number greater than 3) …
  - $p(E_1) = p(\{2\} \cup \{4\} \cup \{6\}) = 3/6$
  - $p(E_2) = p(\{4\} \cup \{5\} \cup \{6\}) = 3/6$
  - $p(E_1 \cap E_2) = p(E_1, E_2) = p(\{4\} \cup \{6\}) = 2/6$
  - Thus the joint probability of E_1 and E_2, $p(E_1, E_2)$, is $1/3$

Aside: Probability

Probabilities of Complements and Unions of Events

For events E_1 and E_2 in a sample space S

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$$

- Example: What is the probability that a positive integer selected at random from the set of positive integers less than or equal to 100 is divisible by either 2 or 5?

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$$

$$= \frac{50}{100} + \frac{20}{100} - \frac{10}{100} = \frac{3}{5}.$$

Aside: Probability

Probabilities of outcomes of experiments where outcomes may **not** be equally likely

- Let S be a sample space of an experiment with a finite or countable number of outcomes

  $p(s)$ is the probability of each outcome s

$$0 \leq p(s) \leq 1 \text{ for each } s \in S$$

$$\sum_{s \in S} p(s) = 1.$$

Aside: Probability

When there are n possible outcomes

$$0 \leq p(x_i) \leq 1 \text{ for } i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} p(x_i) = 1.$$

- The function $p(s)$ or $p(x_i)$ from the set of all outcomes of sample space S is called a probability distribution

- The uniform distribution assigns the probability $1/n$ to each element of S

Aside: Probability

Definition of the probability of an event

$$p(E) = \sum_{s \in E} p(s)$$

The probability of an event E is the sum of the probabilities of the outcomes in E

Aside: Probability

Conditional Probability

– Let E and F be events with $p(F) > 0$

The conditional probability of E given F, denoted $p(E \mid F)$, is defined as

$$p(E \mid F) = \frac{p(E \cap F)}{p(F)} \quad \longleftarrow \quad p(E, F)$$

Aside: Probability

Conditional Probability

– What is the conditional probability that a family with two children has two boys, given that they have at least one boy?

Assume that each of the possibilities BB, BG, GB, GG is equally likely.

Let E be the event that the family with two children as two boys

Let F be the event that a family with two children has at least one boy

$E = \{BB\}$

$F = \{BB, BG, GB\}$

$E \cap F = \{BB\}$

$p(F) = ¾$ and $p(E \cap F) = ¼$

$$p(E \mid F) = \frac{p(E \cap F)}{p(F)} = \frac{1/4}{3/4} = \frac{1}{3}$$

Aside: Probability

Independence

- If $p(E \mid F) = p(E)$ it means F has no bearing on E
- We say E and F are independent events

Definition: the events E and F are independent if and only if

$$p(E \cap F) = p(E)\, p(F)$$

Aside: Probability

Independence

Let E be the event that the family with two children has two boys

Let F be the event that a family with two children has at least one boy

Are the two events independent?

$E = \{BB\}$ so $p(E) = ¼$

$F = \{BB, BG, GB\}$ so $p(F) = ¾$

Thus, $p(E)\ P(F) = 3/16$

$p(E \cap F) = ¼$

Since $p(E \cap F) \neq p(E)\ P(F)$ the events are not independent

Aside: Probability

Random Variables

- Many problems are concerned with a numerical value associated with the outcome of an experiment

  - E.g. the number of 1 bits in a randomly generated string of 10 bits
  - E.g. the number of times a head comes up when you toss a coin 20 times
  - E.g. some feature of a manufactured part

A **random variable** is a **function** from the sample space of an experiment to the real numbers

$$f: S \rightarrow \Re$$

Aside: Probability

Random Variables

- A **random variable** is a **function** from the sample space of an experiment to the real numbers
  - A random variable assigns a real number to each possible outcome
  - The input to a random variable is an elementary outcome, and the output is a number
  - We can think of random variable as numerical measurements of outcomes
  - A random variable is a function
    - It is not a variable
    - It is not random!

Aside: Probability

Random Variables

– For example, toss a coin three times

Let $X(t)$ be the random variable that equals the number of heads that appear when t is the outcome

What are the outcomes? $HHH, HHT, HTH, THH, TTH, THT, HTT, TTT$

$X(HHH) = 3$

$X(HHT) = 2$

$X(HTH) = 2$

$X(THH) = 2$

$X(TTH) = 1$

$X(THT) = 1$

$X(HTT) = 1$

$X(TTT) = 0$

Aside: Probability

Random Variables

The distribution of a random variable X on a sample space S is the set of pairs

$(r, p(X = r))$

For all $r \in X(S)$, where $p(X = r)$ is the probability that X takes the value r

For the previous example,

$p(X = 3) = 1/8$

$p(X = 2) = 3/8$

$p(X = 1) = 3/8$

$p(X = 0) = 1/8$

Hence, the **distribution** of $X(t)$ is the set of pairs

$(3, 1/8), (2, 3/8), (1, 3/8), (0, 1/8)$

Aside: Probability

Bayes' Theorem / Rule

- Shows how to revise probability of events in the light of new data
- For example, we can determine the probability that a particular incoming email is spam using the occurrence of words in the message
- To do this we need to know
  - The percentage of incoming emails that are spam
  - The percentage of **spam** messages in which these words occur
  - The percentage of messages that are **not spam** in which each of these words occur

Aside: Probability

Bayes' Theorem / Rule

Let E be an event from a sample space S

$F_1, F_2, \ldots F_n$, are mutually exclusive events such that

$F_1 \cup F_2, \ldots \cup F_n = S$

$p(E) \neq 0$ and $p(F_i) \neq 0$

Likelihoods

Prior probability

Posterior probability

$$p(F_j \mid E) = \frac{p(E \mid F_j)p(F_j)}{\sum_{i=1}^{n} p(E \mid F_i)p(F_i)}$$

Aside: Probability

Sometimes we write $p(F_j \mid E)$

as $p(H_j \mid D)$

to be read as "The probability that hypothesis H_j is true given data D"

which is computed from the prior probabilities that each hypothesis is true

$$p(H_j)$$

and the conditional probabilities (likelihoods) of that data occurring in the case of each hypothesis

$$p(D \mid H_j)$$

Aside: Probability

See also:

- K. H. Rosen, *Discrete Mathematics and Its Applications*, 2012.
- J. R. Movellan, *Introduction to Probability Theory and Statistics*, 2008.