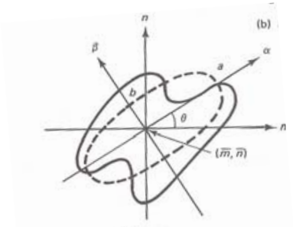
INF 4300 – Digital Image Analysis

OBJECT DESCRIPTION - FEATURE EXTRACTION



(c) Best fit ellipse

Fritz Albregtsen 13.10.2010

Today

We go through

G&W section 11.2 Boundary Descriptors

G&W section 11.3 Regional Descriptors

Curriculum includes these lecture notes.

We cover the following :

1. Introduction
2. Topological features
3. Projections
4. Geometric features
5. Statistical shape features
6. Moment-based geometric features

…

What is feature extraction?

- Devijver and Kittler (1982):
  ”Extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class variability”.
  - Within-class pattern variability: variance between objects belonging to the same class.
    AAAAAAAAAAA
  - Between-class pattern variability: variance between objects from different classes.
    A B C D E F G H I J

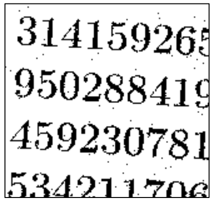
Feature extraction

- We will discriminate between different object classes based on a set of features.
- The features are chosen given the application.
- Normally, a large set of different features is investigated.
- Classifier design also involves feature selection
  - selecting the best subset out of a large feature set.
- Given a training set of a certain size,
  the dimensionality of the feature vector must be limited.
- Careful selection of features is the most important step in image classification!

Describing the shape of a segmented object

Assumptions:

- We have a segmented, labeled image.
- Each object that is to be described has been identified during segmentation.



  - Ideally, one region in the segmented image should correspond to one object.
  - One object should not be fragmented into several non-connected regions.
  - Some small noise objects will often occurr, these can often be removed later.

Example 1: Recognize printed numbers

- Goal: get the series of digits, e.g. 1415926535897……

Steps in the program:

1. Segment the image to find digit pixels.
2. Find angle of rotation and rotate back.
3. Create region objects – one object pr. digit or connected component.
4. Compute features describing shape of objects
5. Train a classifier on many objects of each digit.
6. Classify a new object as the one with the highest probability.

Example 2: Recognize music symbols

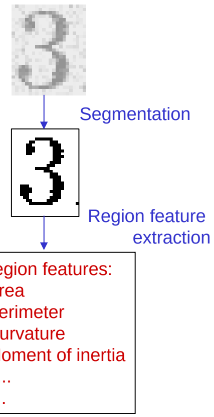
- Goal:interpret the notes and symbols to create a MIDI-file and then play it!

Steps in the program:

1. Segment the image to find symbol pixels.
2. Find angle of rotation and rotate back.
3. Find the note lines and remove them.
4. Create regions objects for connected components.
5. Match each object with known object class (whole note, quarter note, rest, bar, etc.) based on object features.
6. For all notes: find note height given its vertical position.
7. Create a midi file from this.

From pixels to features

- Input to the classifier is normally a set of features derived from the image data, not the image data itself.
- Why can't we just use all the gray level pixels as they are for text recognition?
  - Objects corresponds to regions. We need the spatial relationship between the pixels.
  - For text recogntion: the information is in shape, not in gray levels.



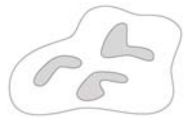
Typical image analysis tasks

- Preprocessing/noise filtering
- Segmentation
- Feature extraction
  - Are the original image pixel values sufficient for classification, or do we need additional features?
  - What kind of features do we use in order to discriminate between the object classes involved?
- Exploratory feature analysis and selection
  - Which features separate the object classes best?
  - How many features are needed?
- Classification
  - From a set of object examples with known class, decide on a method that separates objects of different types.
  - For new objects: assign each object/pixel to the class with the highest probability
- Validation of classifier accuracy

Topologic features

- This is a group of invariant integer features
  - Invariant to position, rotation, scaling, warping
- Features based on the object skeleton
  - Number of terminations (one line from a point)
  - Number of breakpoints or corners (two lines from a point)
  - Number of branching points (three lines from a point)
  - Number of crossings (> three lines from a point)
- Region features:
  - Number of holes in the object (H)
  - Number of components (C)
  - Euler number, E = C - H
    - Number of connected components – number of holes
  - Symmetry

Region with two holes



Regions with three connected components

1D Projections

- For each row in the region, count the number of object pixels.

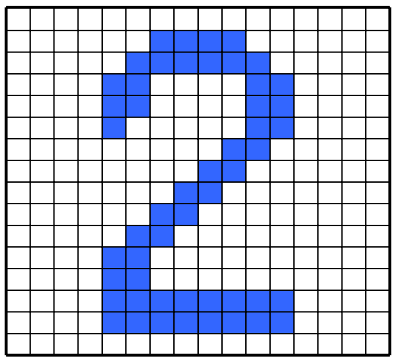
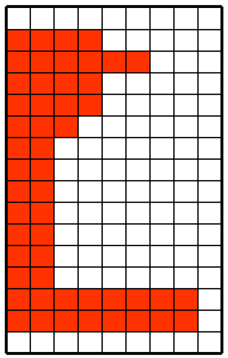


Image – binary region pixels



Row histogram

Projections

- 1D horizontal projection of the region:

$$p_h(x) = \sum_y f(x, y)$$

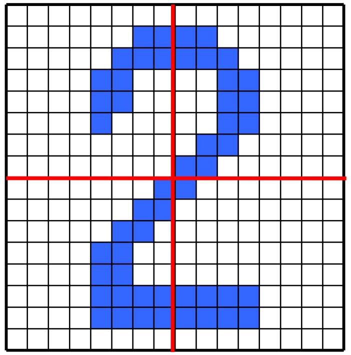
- 1D vertical projection of the region:

$$p_v(y) = \sum_x f(x, y)$$

- Can be made scale independent by using a fixed number of bins and normalizing the histograms.
- Radial projection in reference to centroid -> ”signature”.

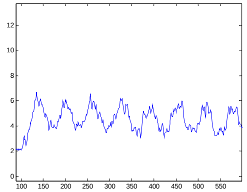
Use of projection histograms

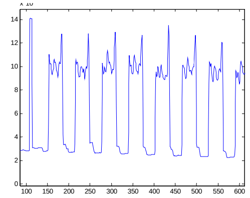
- Divide the object into different regions and compute projection histograms for each region.
  - How can we use this to separate 6 and 9?
- The histograms can also be used as features directly.



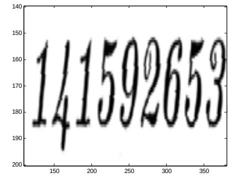
Use of projection histograms

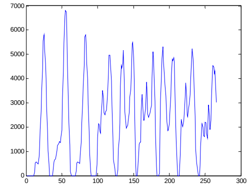
- Check if a page with text is rotated





- Detecting lines, connected objects or single symbols





Geometric features from contours

- Boundary length/perimeter
- Area
- Curvature
- Diameter/major/minor axis
- Eccentricity
- Bending energy
- Basis expansion (Fourier – last week)

Perimeter length from chain code

- Distance measure differs when using 8- or 4-neighborhood
- Using 4-neighborhood, measured length ≥ actual length.
- In 8-neighborhood, fair approximation from chain code by:

$$P_F = n_{even} + n_{odd}\sqrt{2}$$

- This overestimates real perimeters systematically.
- A general correction factor of 0.95 works satisfactory.
- For straight lines, find a corner count $\mathbf{n_c}$ as the number of occurrences of consecutive unequal chain elements, then:

$$P_{VS} = 0.980\,n_{even} + 1.406\,n_{odd} - 0.091\,n_c$$

- For general blob-like objects:

$$P_K = \pi\left(1+\sqrt{2}\right)\left(n_{even} + n_{odd}\sqrt{2}\right)/8$$

Area of region

- Straight-forward implementation: traverse all object pixels.

- Can also be calculated from the boundary by Greens theorem
- Surface integral equals boundary integral

$$A = \int\int_S dxdy = \int_C xdy$$

- Simple to implement:
  - follow contour, x and dy follow from pixels in the sequence

- Approximate area from (x,y)-coordinates of N polygon vertices:

$$\hat{A} = \frac{1}{2}\left|\sum_{k=0}^{N-1}(x_k y_{k+1} - y_k x_{k+1})\right|$$

- Where sign of sum indicates clockwise or anti-clockwise direction.

Compactness and circularity

- Compactness (very simple measure)
  - $\gamma = P^2/(4\pi A)$, where P = Perimeter, A = Area,
  - For a circular disc, γ is minimum and equals 1.
  - Compactness attains high value for complex object shapes, but also for very elongated simple objects, like rectangles and ellipses where a/b ratio is high.

- G&W defines
  - Compactness = P^2/A
  - Circularity ratio = $4\pi A/P^2$

Curvature

- In the continous case, curvature is the rate of change of slope.

$$|\kappa(s)|^2 = \left[\frac{d^2x}{ds^2}\right]^2 + \left[\frac{d^2y}{ds^2}\right]^2$$

- In the discrete case, difficult because boundary is locally ragged.

- Use difference between slopes of adjacent boundary segments to describe curvature at point of segment intersection.

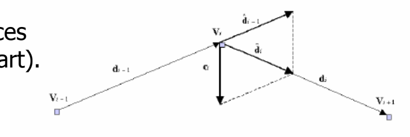
- Curvature can be calculated from chain code.

Discrete computation of curvature

- Trace the boundary and insert vertices at a given distance (e.g. 3 pixels apart).

- Compute local curvature c_i as the difference between the directions of two edge segments joining a vertex:

$$c_i = \vec{d}_i - \vec{d}_{i-1}$$

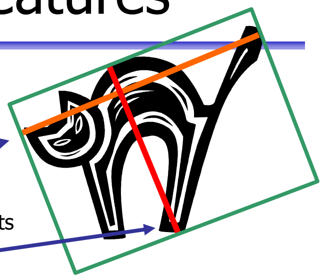


v_i: edge segment i
$d\hat{}_{i-1}$: unit vectors of edge segments d_{i-1} and dt
c_i: local curvature at point i

- Curvature feature: sum all local curvature measures along the border.

- More complex regions get higher curvature.

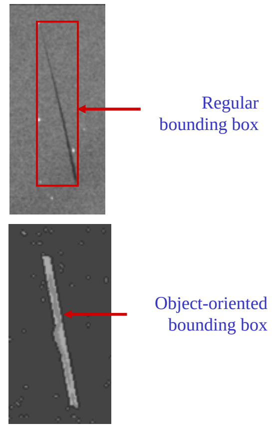
Contour based features



- Diameter = Major axis (a)
  Longest distance of a line segment connecting two points on the perimeter
- Minor axis (b)
  Computed along a direction perpendicular to the major axis. Largest length possible between two border points in the given direction.
- "Eccentricity" of the contour (a/b)

Bounding box and CH features

- Regular bounding box
  - Width/height of bounding box
  - Centre of mass position in box
- If the object's orientation is known, a bounding box can also be oriented along this direction.
- Extent = Area/(Area of bounding box)
- Solidity = Area/(Area of Convex Hull)



Moments

- Borrows ideas from physics and statistics.
- For a given continuous intensity distribution *g(x, y)* we define moments m_{pq} by

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q g(x,y)\,dx\,dy$$

- For sampled (and bounded) intensity distributions f (x, y)

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

- A moment m_{pq} is of *order* p + q.

Moments from binary images

- For binary images, where
  f (x, y) = 1 ⇒ object pixel
  f (x, y) = 0 ⇒ background pixel
- Area

$$m_{00} = \sum_x \sum_y f(x,y)$$

- Center of mass /"tyngdepunkt"

$$m_{10} = \sum_x \sum_y x f(x,y) = \bar{x} m_{00} \quad \Rightarrow \quad \bar{x} = \frac{m_{10}}{m_{00}}$$

$$m_{01} = \sum_x \sum_y y f(x,y) = \bar{y} m_{00} \quad \Rightarrow \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

Grayscale moments

- In gray scale images, we may regard *f(x,y)* as a discrete 2-D probability distribution over (x,y)
- For probability distributions, we should have

$$m_{00} = \sum_x \sum_y f(x,y) = 1$$

  - And if this is not the case we can normalize by

$$F(x,y) = f(x,y)/m_{00}$$

Central moments

- These are position invariant moments

$$\mu_{p,q} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

- where

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

- The total mass and the center of mass are given by

$$\mu_{00} = \sum_x \sum_y f(x,y), \qquad \mu_{10} = \mu_{01} = 0$$

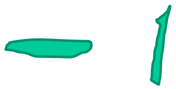
- This corresponds to computing ordinary moments after having translated the object so that center of mass is in origo.
- Central moments are independent of position, but are not scaling or rotation invariant.
- What is μ_{00} for a binary object?

Moments of inertia or Variance

- The two second order central moments measure the spread of points around the y- and x-axis through the centre of mass

$$\mu_{20} = \sum_x \sum_y (x-\bar{x})^2 f(x,y)$$

$$\mu_{02} = \sum_x \sum_y (y-\bar{y})^2 f(x,y)$$



- From physics: moment of inertia about an axis: how much energy is required to rotate the object about this axis:
  - Statisticans like to call this variance.
- The cross moment of intertia is given by

$$\mu_{11} = \sum_x \sum_y (x-\bar{x})(y-\bar{y}) f(x,y)$$

  - statisticians call this covariance or correlation.
- Orientation of the object can be derived from these moments.
  - This implies that they are not invariant to rotation.

Moments of an ellipse

- Assume that the ellipse has semimajor and semiminor axes (*a,b*). For an ellipse given by

$$(x/a)^2 + (y/b)^2 = 1 \quad \Rightarrow y = \pm \frac{b}{a}\sqrt{a^2 - x^2}$$

  the largest second order central moment is given by

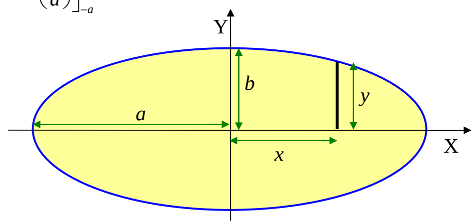
$$I_{20} = 2\frac{b}{a}\int_{-a}^{a} x^2 \sqrt{a^2 - x^2}\,dx$$

$$I_{20} = 2\frac{b}{a}\left[\frac{x}{8}\left(2x^2 - a^2\right)\sqrt{a^2 - x^2} + \frac{a^4}{8}\sin^{-1}\left(\frac{x}{a}\right)\right]_{-a}^{a}$$

$$I_{20} = 2\frac{b}{a}\left[\frac{a^4}{8}\left(\frac{\pi}{2} + \frac{\pi}{2}\right)\right] = \underline{\underline{\frac{\pi}{4}a^3 b}}$$

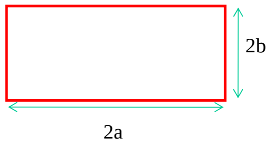
The smallest moment of inertia is

$$I_{\min} = \underline{\underline{\frac{\pi}{4}a\, b^3}}$$

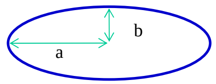


Moments of inertia for simple shapes

- Rectangular object (2a×2b):
  $I_{20} = 4a^3b/3 \quad , \quad I_{02} = 4ab^3/3$



- Square (a×a):
  $I_{20} = I_{02} = a^4/12$
- Elliptical object, semi-axes (a,b):
  $I_{20} = \pi a^3 b/4 \; , \quad I_{02} = \pi ab^3/4$



- Circular object, radius R:
  $I_{20} = I_{02} = \pi R^4/4$

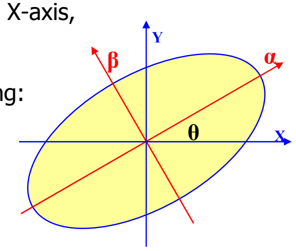
Object orientation - I

- Orientation is defined as the angle, relative to the X-axis, of an axis through the centre of mass that gives the lowest moment of inertia.
- Orientation θ relative to X-axis found by minimizing:

$$I(\theta) = \sum_{\alpha}\sum_{\beta} \beta^2 \; f(\alpha,\beta)$$

where the rotated coordinates are given by

$$\alpha = x\cos\theta + y\sin\theta, \quad \beta = -x\sin\theta + y\cos\theta$$



- The second order central moment of the object around the α-axis, expressed in terms of x, y, and the orientation angle θ of the object:

$$I(\theta) = \sum_{x}\sum_{y} [y\cos\theta - x\sin\theta]^2 \; f(x,y)$$

- We take the derivative of this expression with respect to the angle θ
- Set derivative equal to zero, and find a simple expression for θ :

Object orientation - II

- Second order central moment around the α-axis:

$$I(\theta) = \sum_{x}\sum_{y} [y\cos\theta - x\sin\theta]^2 \; f(x,y)$$

- Derivative w.r.t. Θ = 0 =>

$$\frac{\partial}{\partial\theta} I(\theta) = \sum_{x}\sum_{y} 2f(x,y)[y\cos\theta - x\sin\theta][-y\sin\theta - x\cos\theta] = 0$$

$$\Downarrow$$

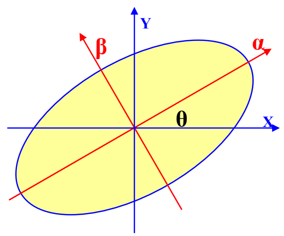
$$\sum_{x}\sum_{y} 2f(x,y)\left[xy\left(\cos^2\theta - \sin^2\theta\right)\right] = \sum_{x}\sum_{y} 2f(x,y)\left[x^2 - y^2\right]\sin\theta\cos\theta$$

$$\Downarrow$$

$$2\mu_{11}\left(\cos^2\theta - \sin^2\theta\right) = 2(\mu_{20} - \mu_{02})\sin\theta\cos\theta$$

$$\Downarrow$$

$$\frac{2\mu_{11}}{(\mu_{20} - \mu_{02})} = \frac{2\sin\theta\cos\theta}{(\cos^2\theta - \sin^2\theta)} = \frac{2\tan\theta}{1 - \tan^2\theta} = \tan(2\theta)$$

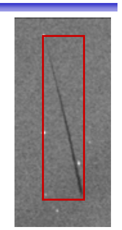


- So the object orientation is given by:

$$\underline{\underline{\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{(\mu_{20} - \mu_{02})}\right]}}, \qquad \textit{where} \;\; \theta \in \left[0, \pi/2\right] \textit{if } \mu_{11} > 0, \;\; \theta \in \left[\pi/2, \pi\right] \textit{if } \mu_{11} < 0$$

Bounding box - again

- Image-oriented bounding box:
  - The smallest rectangle around the object, having sides parallell to the edges of the image.
  - Found by searching for min and max x and y within the object (*xmin, ymin, xmax, ymax*)



- Object-oriented bounding box:
  - Smalles rectangle around the object, having one side parallell to the orientation of the object (θ).
  - The transformation

    $$\alpha = x\cos\theta + y\sin\theta, \quad \beta = y\cos\theta - x\sin\theta$$

    is applied to all pixels in the object (or its boundary).
  - Then search for α_{min}, β_{min}, α_{max}, β_{max}

The best fitting ellipse

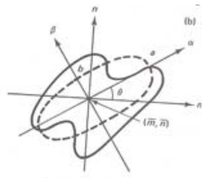
- Object ellipse is defined as the ellipse whose least and greatest moments of inertia equal those of the object.
- Semi-major and semi-minor axes are given by

$$(\hat{a},\hat{b}) = \sqrt{\frac{2\left[\mu_{20}+\mu_{02} \pm \sqrt{(\mu_{20}+\mu_{02})^2 + 4\mu_{11}^2}\right]}{\mu_{00}}}$$

- Numerical eccentricity is given by

$$\hat{\varepsilon} = \sqrt{\frac{\hat{a}^2 - \hat{b}^2}{\hat{a}^2}}$$

- Orientation invariant object features.
- Gray scale or binary object.


(c) Best fit ellipse

What if we want scale-invariance?

- Changing the scale of *f(x,y)* by (α,β) gives a new image:

$$f'(x,y) = f\left(x/\alpha, y/\beta\right)$$

- The transformed central moments

$$\mu'_{pq} = \alpha^{1+p}\beta^{1+q}\mu_{pq}$$

- If $\alpha=\beta$, scale-invariant central moments are given by the normalization:

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}}, \quad \gamma = \frac{p+q}{2}+1, \quad p+q \geq 2$$

Hu's moments: A set of 7 moments that are invariant to translation, scaling and rotation

For second order moments (p+q=2), two invariants are used:

$\varphi_1 = \eta_{20} + \eta_{02}$

$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$

For third order moments, (p+q=3), we can use

$a = (\eta_{30} - 3\eta_{12})$, $b = (3\eta_{21} - \eta_{03})$,

$c = (\eta_{30} + \eta_{12})$, and $d = (\eta_{21} + \eta_{03})$

and simplify the five last invariants of the set:

$\varphi_3 = a^2 + b^2$

$\varphi_4 = c^2 + d^2$

$\varphi_5 = ac[c^2 - 3d^2] + bd[3c^2 - d^2]$

$\varphi_6 = (\eta_{20} - \eta_{02})[c^2 - d^2] + 4\eta_{11}cd$

$\varphi_7 = bc[c^2 - 3d^2] - ad[3c^2 - d^2]$

Moments as shape features

- The central moments are seldom used directly as shape descriptors.
- Major and minor axis are useful shape descriptors.
- Object orientation is normally not used directly, but to estimate rotation.
- The set of 7 Hu moments can be used as shape features. (Start with the first four, as the last half are often zero for simple objects).

Moments that are invariant to general affine transforms

$$
\begin{aligned}
I_1 &= \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4} \\
I_2 &= \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3\mu_{03} - 3\mu_{12}^2\mu_{21}^2}{\mu_{00}^{10}} \\
I_3 &= \frac{\mu_{20}(\mu_{21}\mu_{30} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12} + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7} \\
I_4 &= \{\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 \\
&\quad +12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} \\
&\quad -8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21}^2 \\
&\quad +12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2\}/\mu_{00}^{11}
\end{aligned}
$$

Contrast invariants

- Abo-Zaid *et al.* have defined a normalization that cancels both scaling and contrast.
- The normalization is given by

$$\eta'_{pq} = \frac{\mu_{pq}}{\mu_{00}}\left(\frac{\mu_{00}}{\mu_{20}+\mu_{02}}\right)^{\frac{(p+q)}{2}}$$

- This normalization also reduces the dynamic range of the moment features, so that we may use higher order moments without having to resort to logarithmic representation.
- Abo-Zaid's normalization cancels the effect of changes in contrast, but not the effect of changes in intensity:

$$f'(x,y) = f(x,y) + b$$

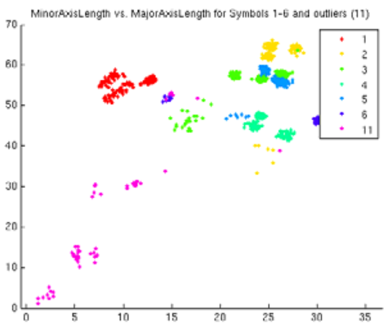
- In practice, we often experience a combination:

$$f'(x,y) = cf(x,y) + b$$

Scatter plots

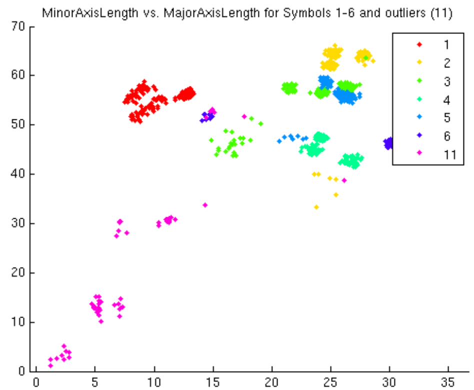
- A 2D scatter plot is a plot of feature values for two different features. Each object's feature values are plotted in the position given by the features values, and with a class label telling its object class.
- Matlab: gscatter(feature1, feature2, labelvector)
- Classification is done based on more than two features, but this is difficult to visualize.
- Features with good class separation show clusters for each class, but different clusters should ideally be separated.

Feature 2: major axis length



Feature 1: minor axis length

Which numbers are well and bad separated?



Two correlated features