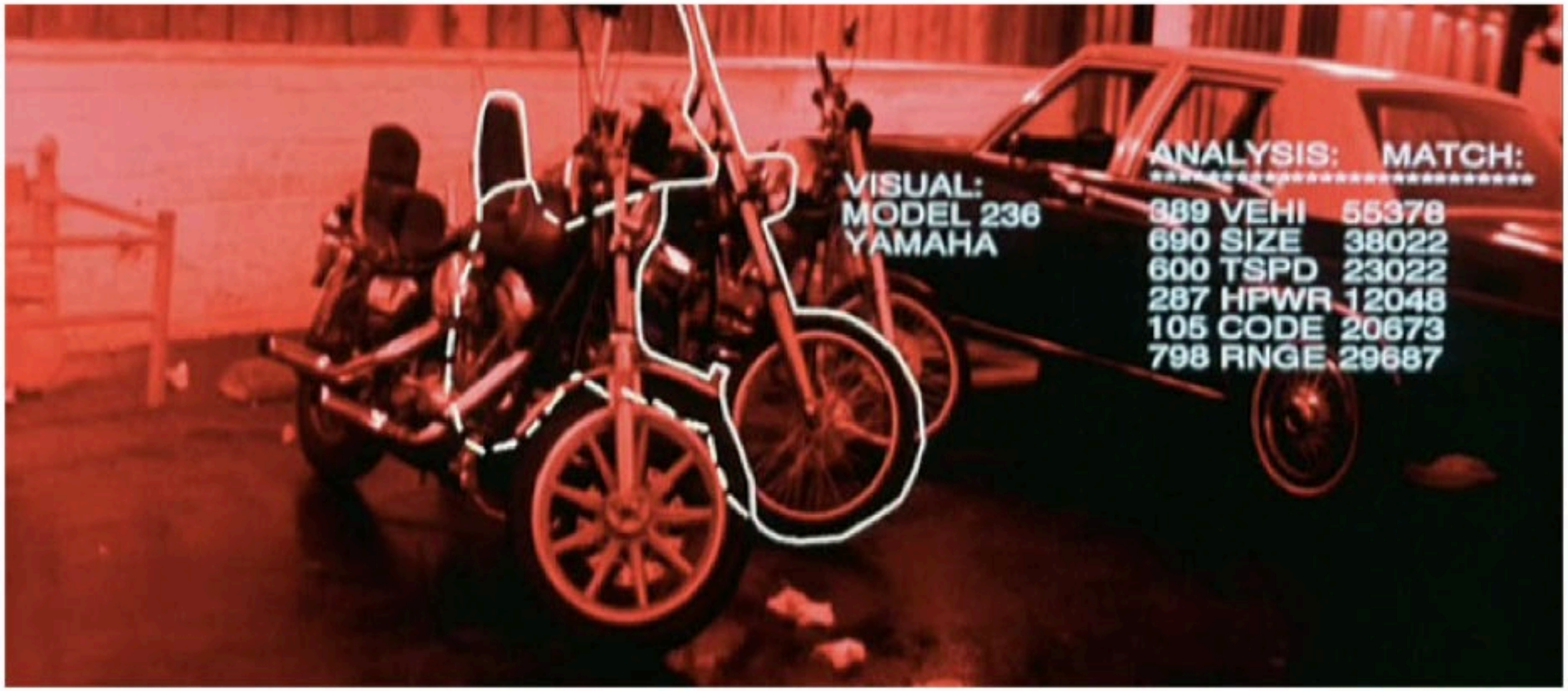


Computer Vision

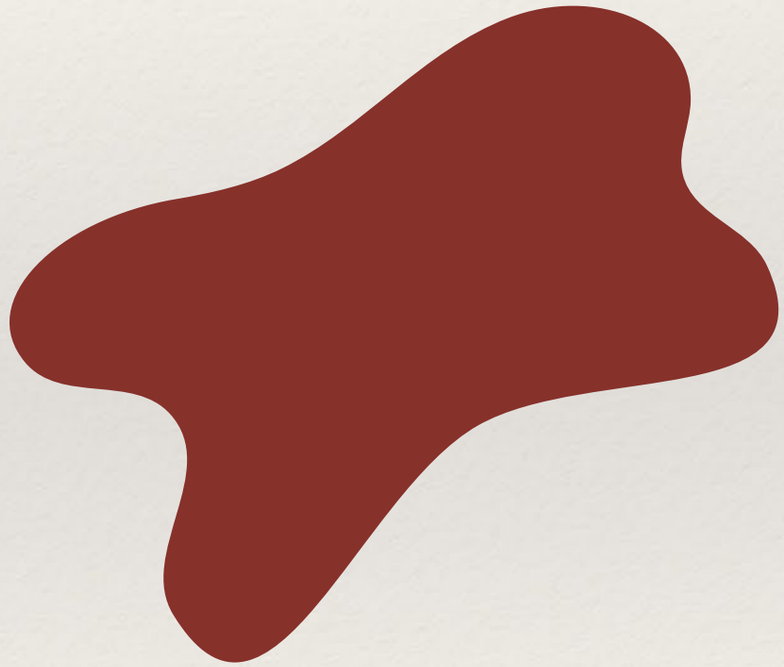
Shape description and modelling

Hansung Kim
h.kim@soton.ac.uk

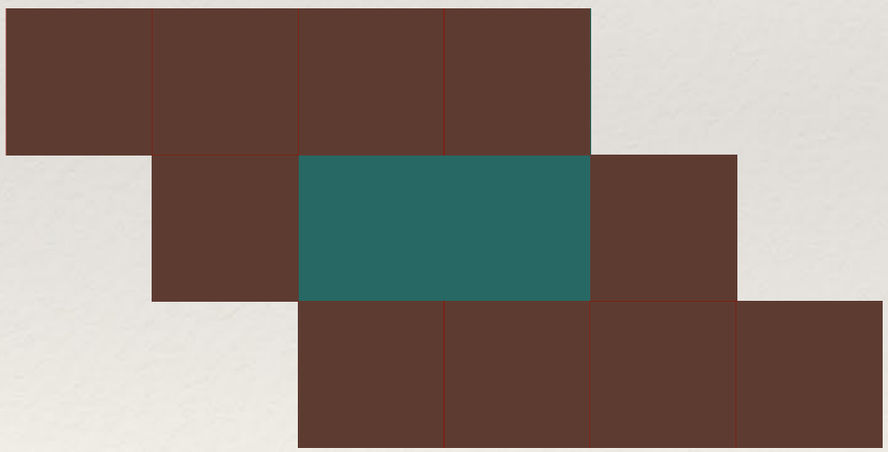
Extracting features from shapes represented by connected components

Recap: Connected Component

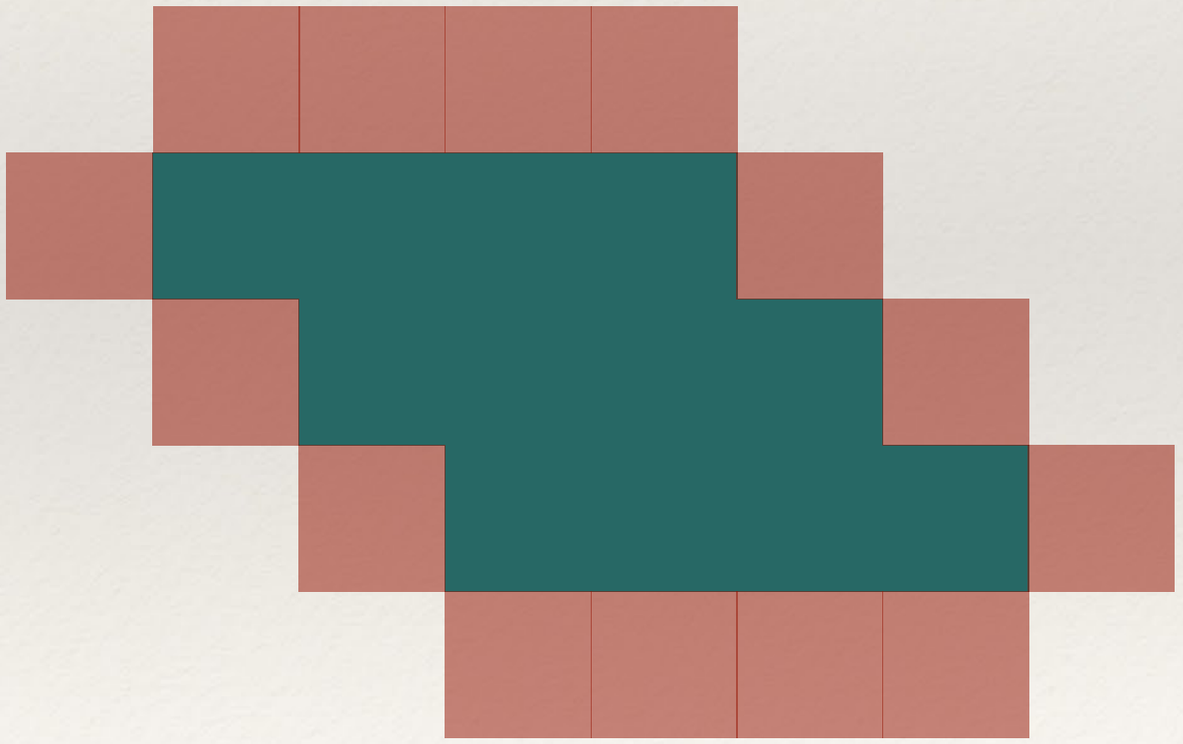
A connected component is a set of pixels in which every pixel is connected either directly or through any connected path of pixels from the set.



Borders

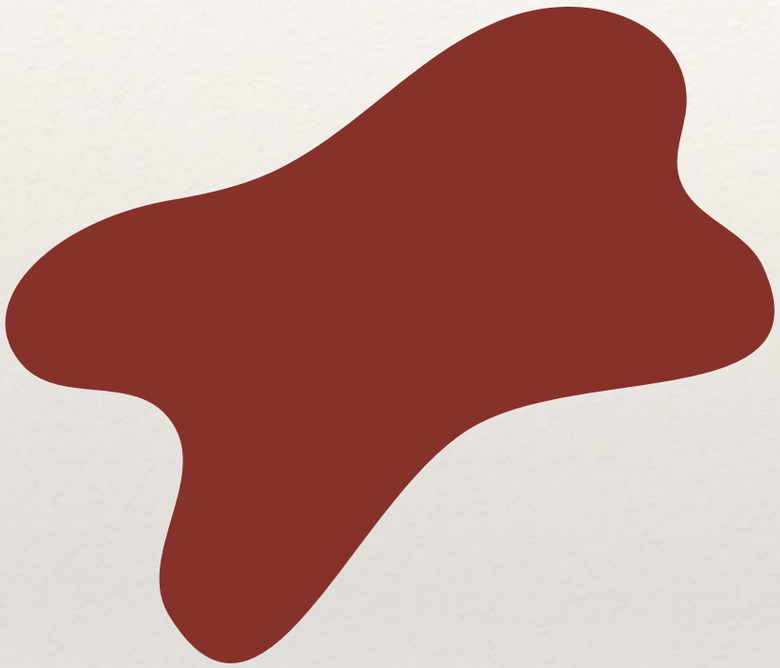


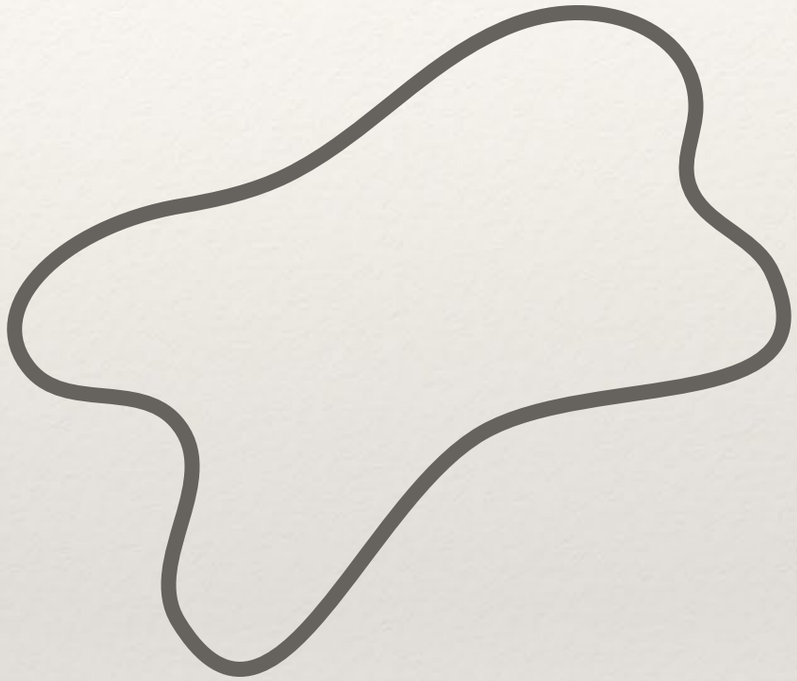
Inner Border



Outer Border

Two ways to describe shape





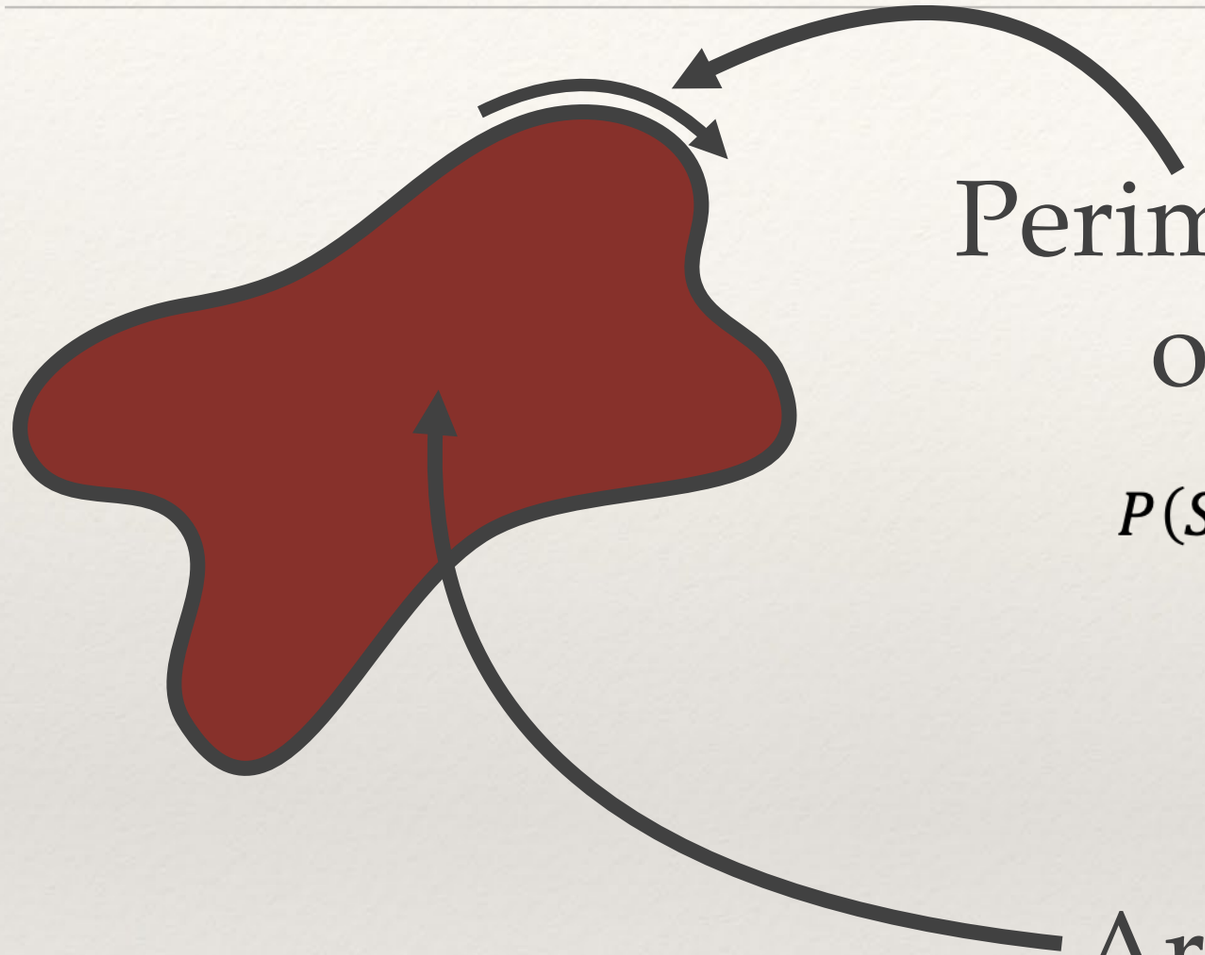
Region Description

Boundary Description

Region Description

1. Simple Scalar Shape Features

Area and Perimeter



Perimeter = length around the outside of component

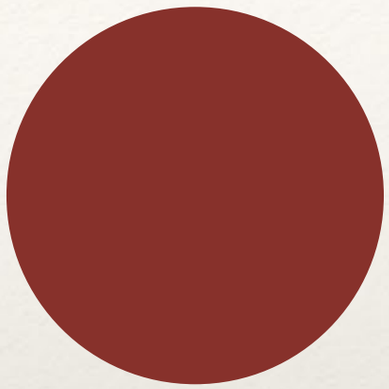
$$P(S) = \sum_j \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}$$

Area = number of pixels in component

Compactness

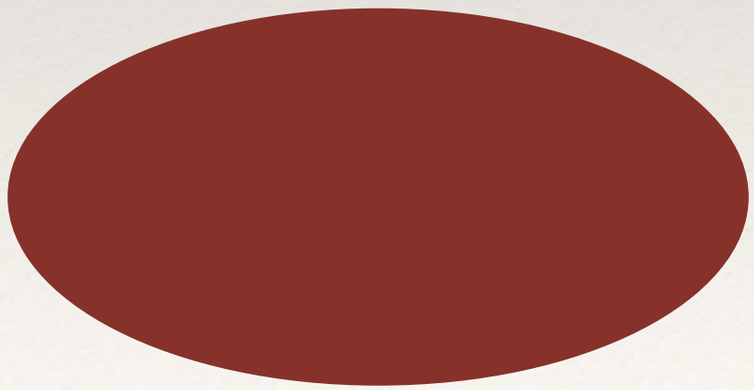
- Compactness measures how tightly packed the pixels in the component are.
- It's often computed as the weighted ratio of area to perimeter squared:

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



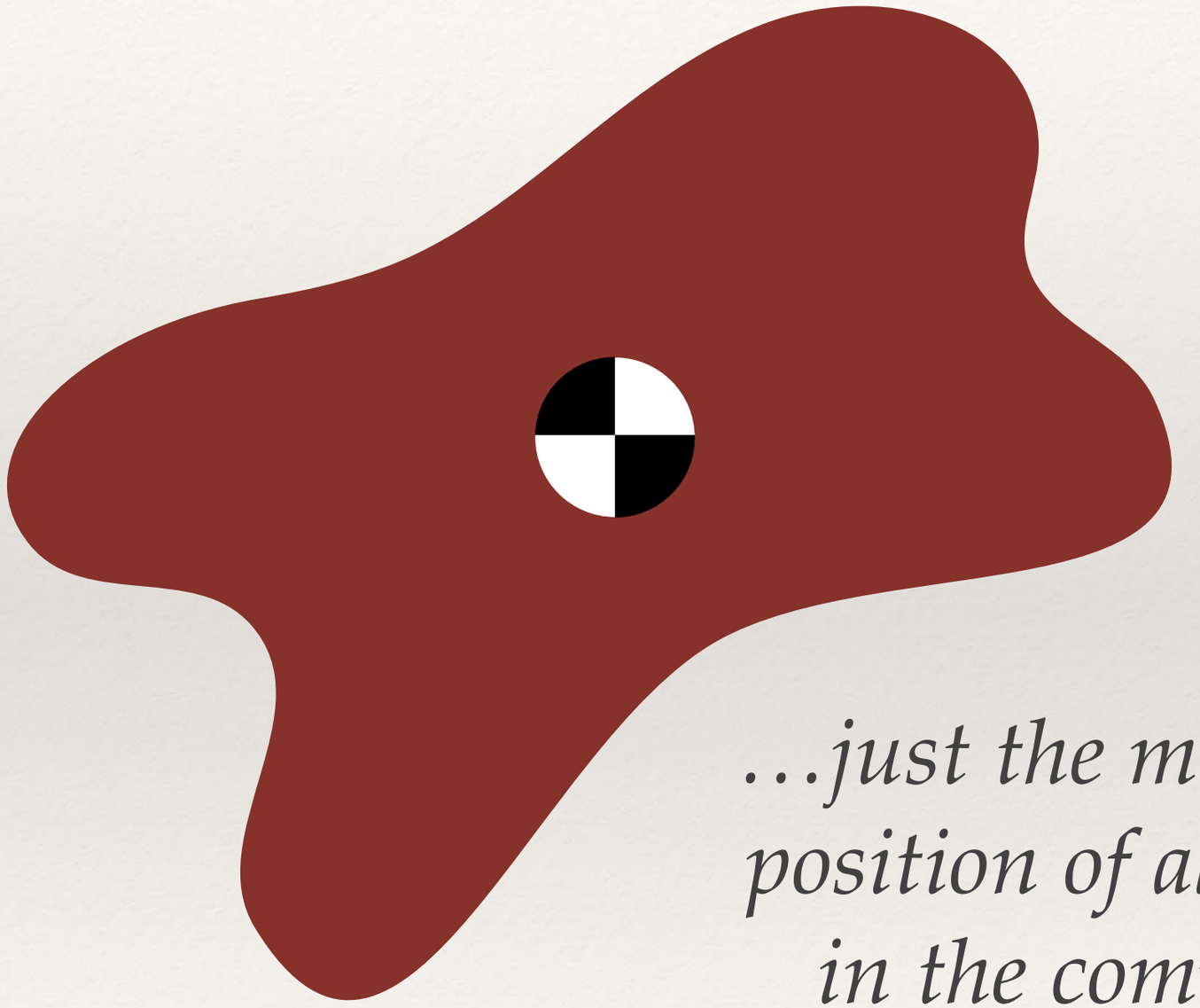
C≅1

C≅0.11



C≅0.79

Centre of Mass

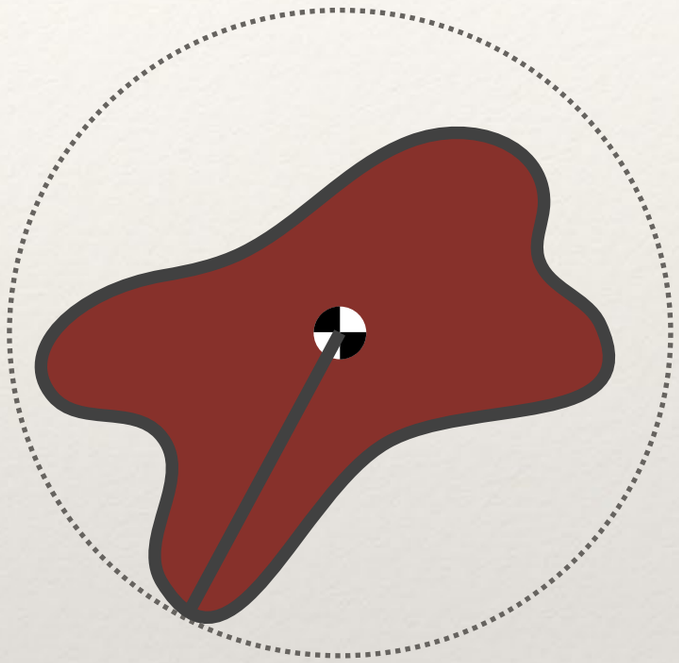


…just the mean x and y position of all the pixels in the component…

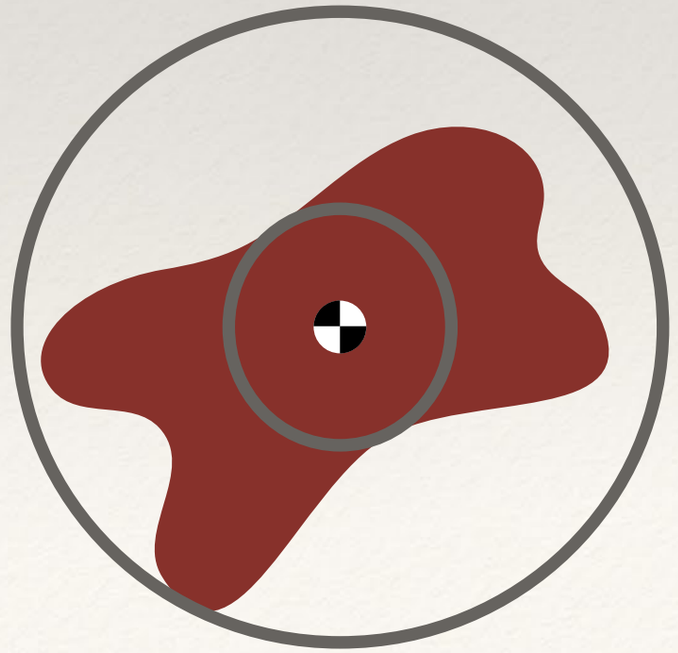
Irregularity/dispersion

A measure of how ‘spread-out’ the shape is

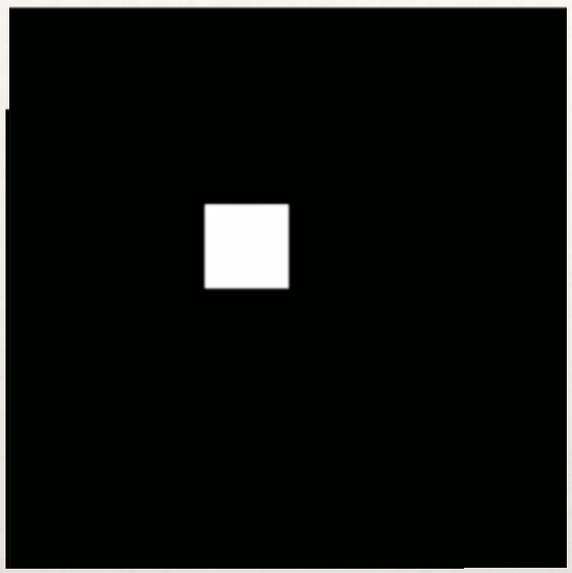
$$I(s) = \frac{\pi \max\left(\left(x_j - \bar{x}\right)^2 + \left(y_j - \bar{y}\right)^2\right)}{A(s)}$$



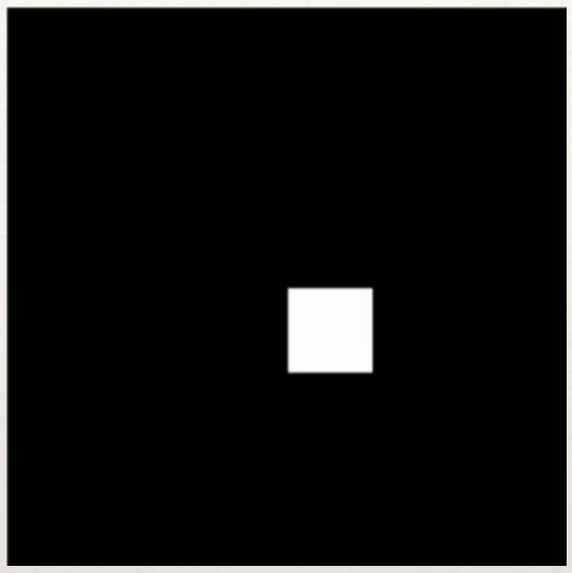
$$IR(s) = \frac{\max\left(\sqrt{\left(x_j - \bar{x}\right)^2 + \left(y_j - \bar{y}\right)^2}\right)}{\min\left(\sqrt{\left(x_j - \bar{x}\right)^2 + \left(y_j - \bar{y}\right)^2}\right)}$$



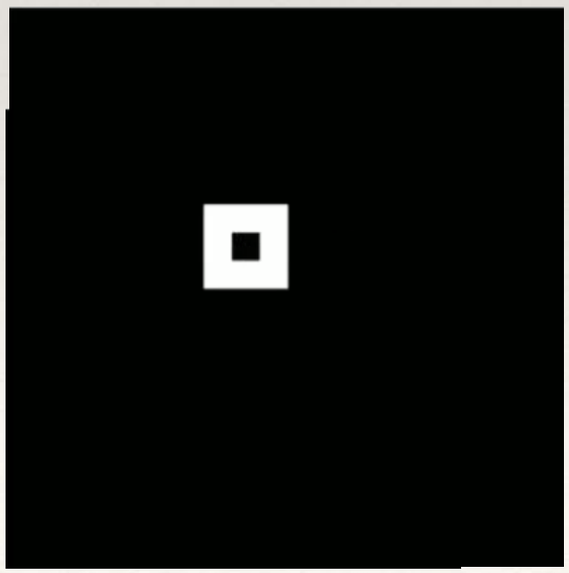
Scalar region features



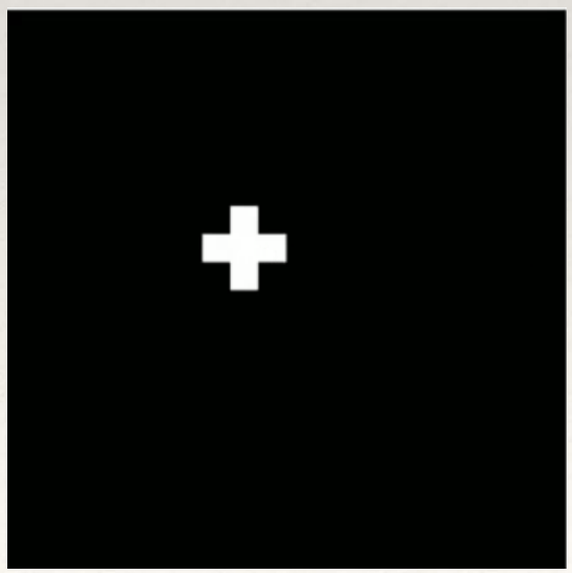
Area: 9.0
Perimeter: 12
Compactness: 0.79
Irregularity I(s): 1.75
Irregularity IR(s): 1.12



Invariant to rotation and translation



Area: 8.0
Perimeter: 12
Compactness: 0.70
Irregularity I(s): 1.96
Irregularity IR(s): 1.12



Area: 5.0
Perimeter: 12
Compactness: 0.43
Irregularity I(s): 2.51
Irregularity IR(s): 1.41

2. Moments

Standard Moments

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

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

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

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

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

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

Central Moments

- Standard 2D moments can be used as shape descriptors
  - But, they're not invariant to translation, rotation and scaling
- **Central Moments** are computed about the centroid of the shape, and are thus translation invariant:

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

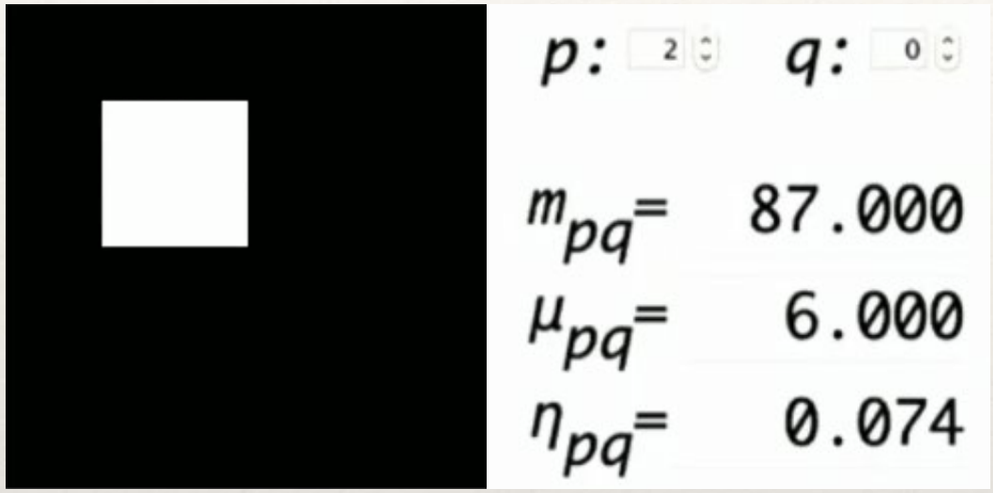
- Note: μ_{01} and μ_{10} are always 0, so have no descriptive power!

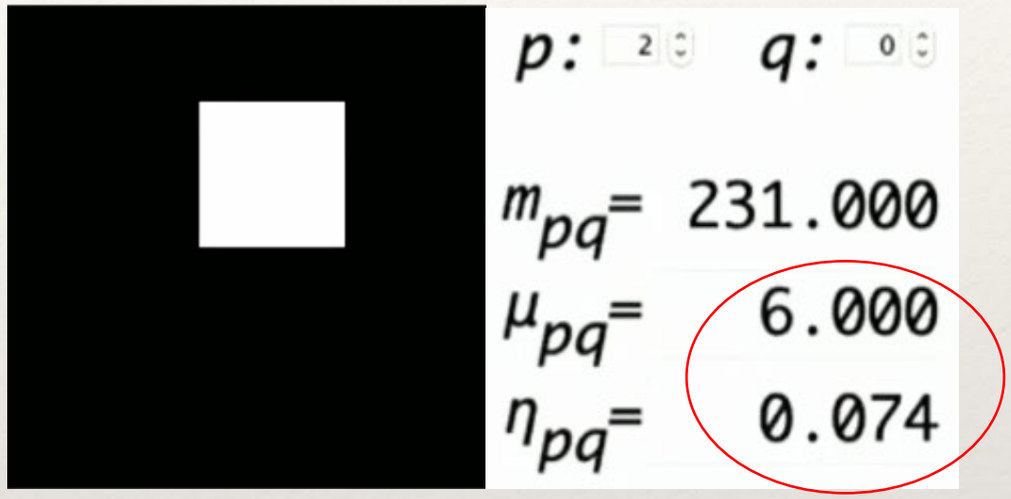
Normalised Central Moments

- **Normalised Central Moments** are both scale and translation invariant:

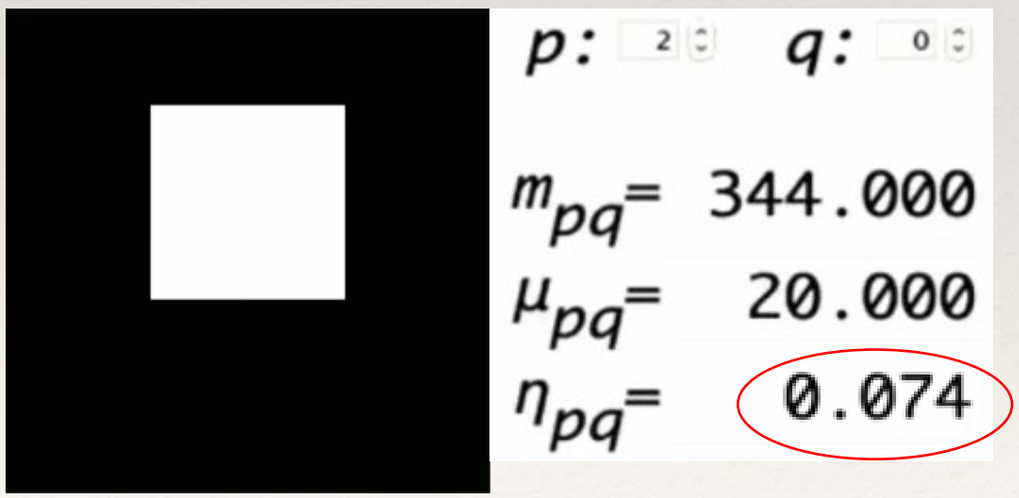
$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \text{ where } \gamma = \frac{(p+q)}{2} + 1$$

Moments



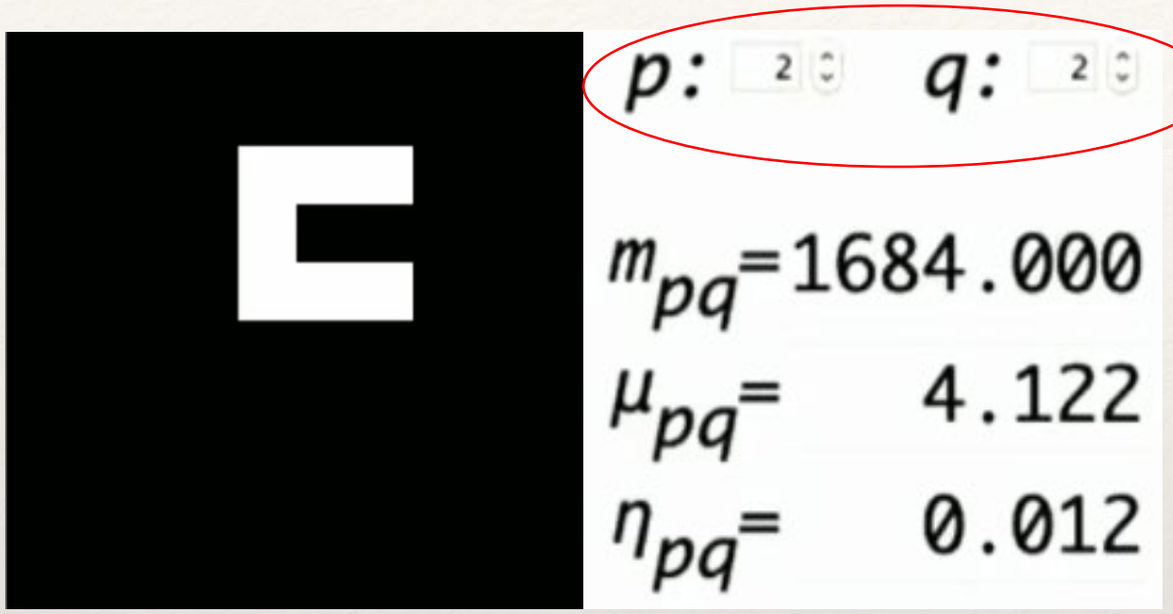


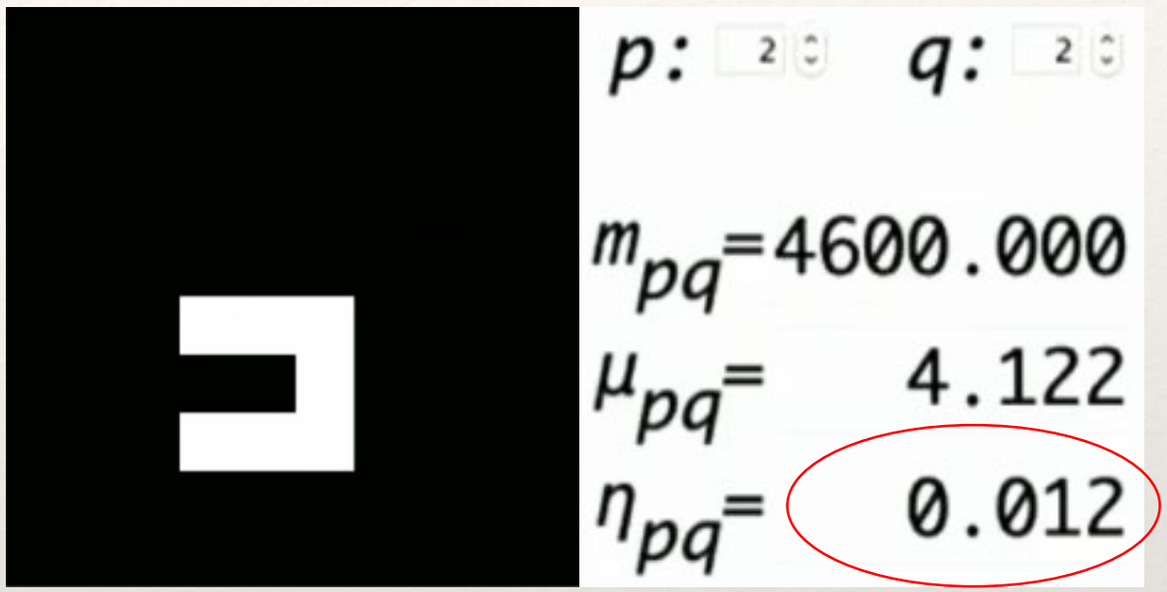
Translation



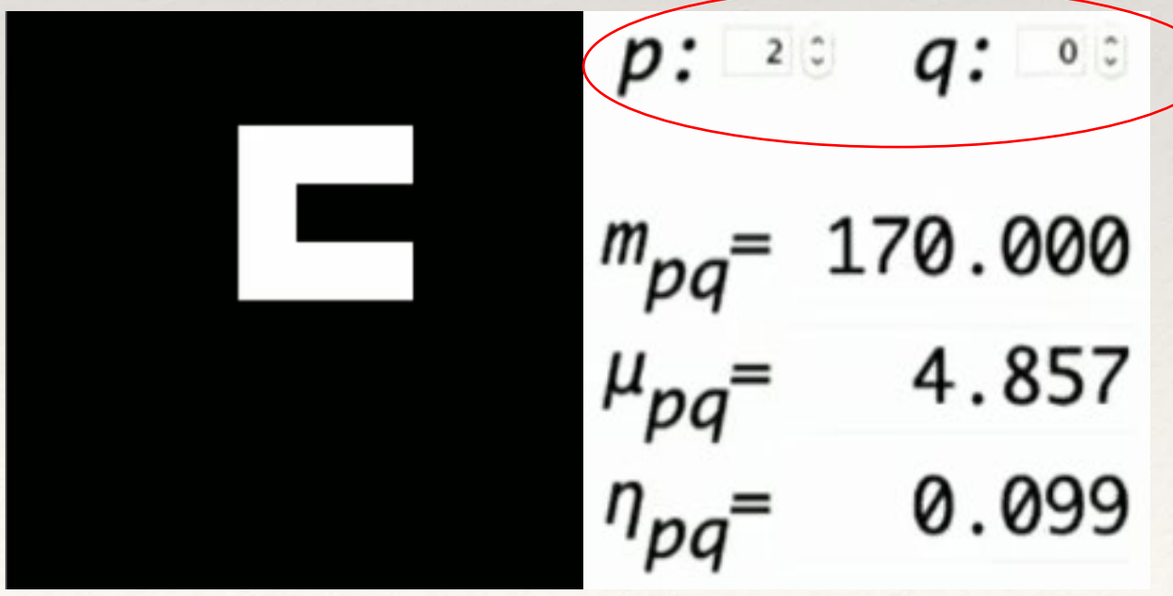
Scaling

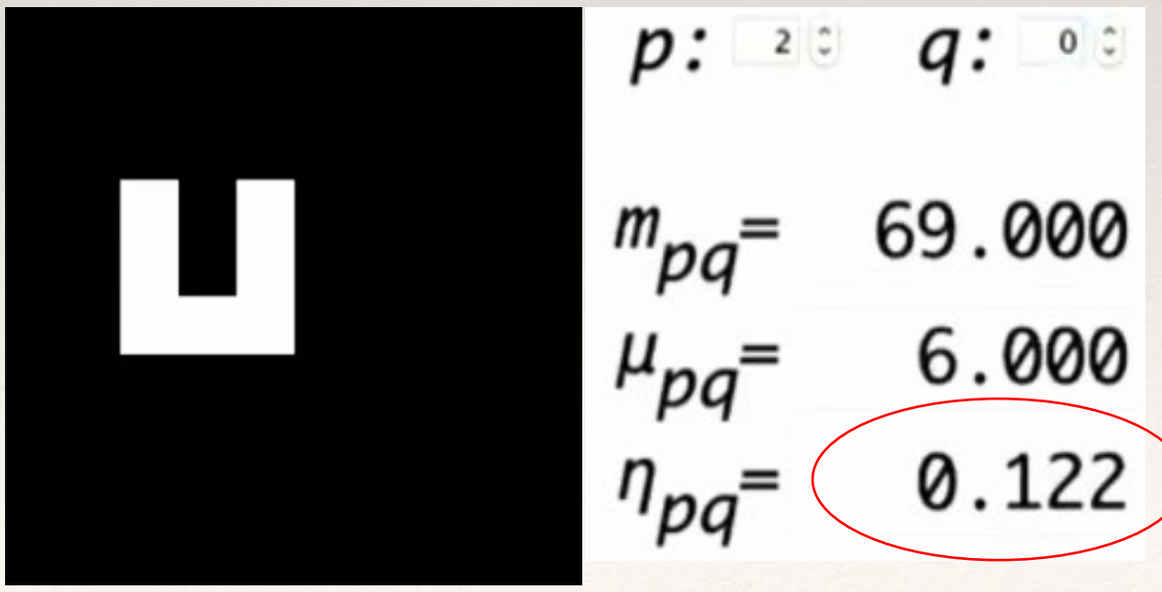
Moments





Rotation invariant?





Hu Moments

- Hu Moments are a set of 7 translation, scale and rotation invariant moments:

$$
\begin{aligned}
M1 &= \eta_{20} + \eta_{02} \\
M2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
M3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
M4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
M6 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
M7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&\quad + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{12} + \eta_{30})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned}
$$

Hu Moments

Translation

Scaling

Rotation

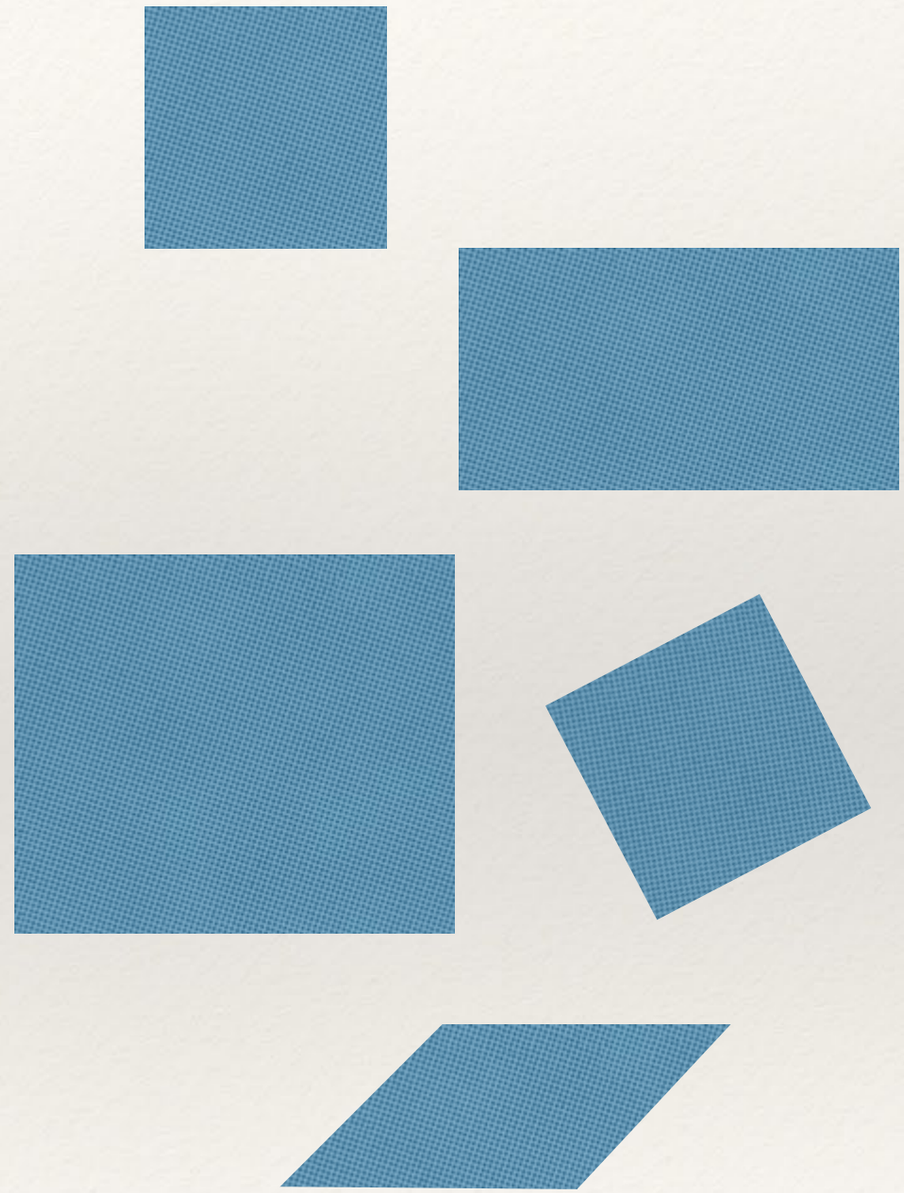
Flip

id	Image	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
K0		2.78871	6.50638	9.44249	9.84018	-19.593	-13.1205	19.6797
S0		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S1		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S2		2.65884	5.7358	9.66822	10.7427	-20.9914	-13.8694	21.3202
S3		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	21.8214
S4		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	-21.8214

https://learnopencv.com/shape-matching-using-hu-moments-c-python/

Even more invariance

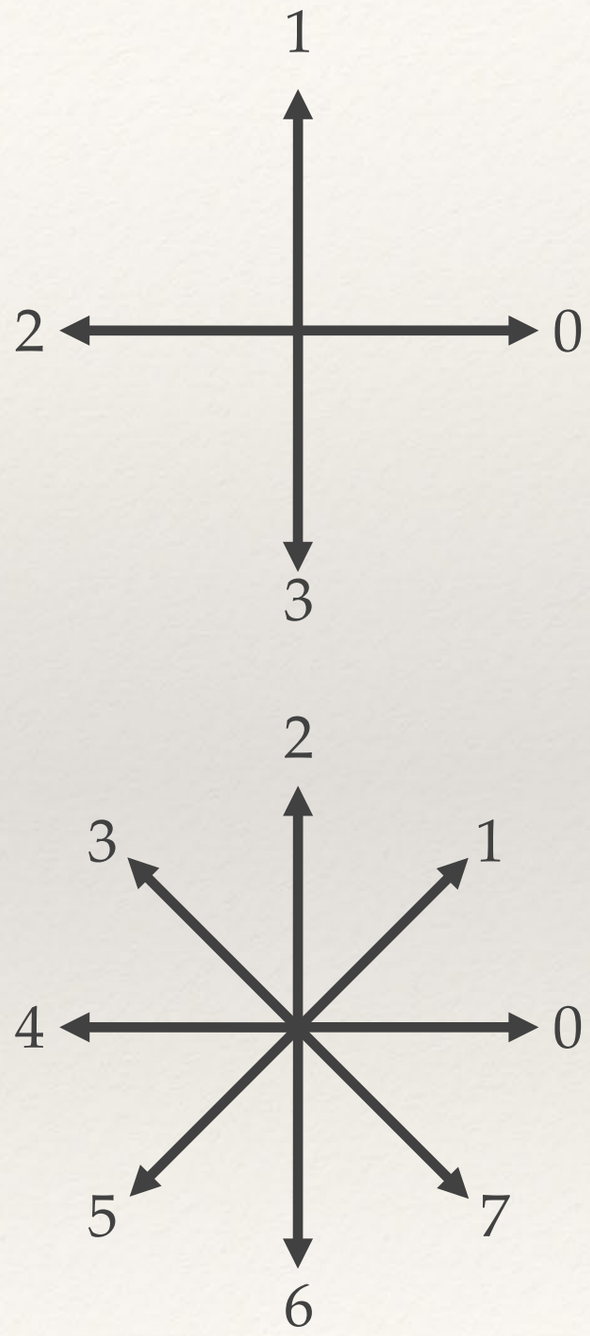
- There's another common set of rotation invariant moments called *Zernike moments*.
  - Used a lot in medical imaging for describing the shape of tumours.
- Other sets moments can be even more invariant to shape transformation
  - e.g. *Affine invariant moments*



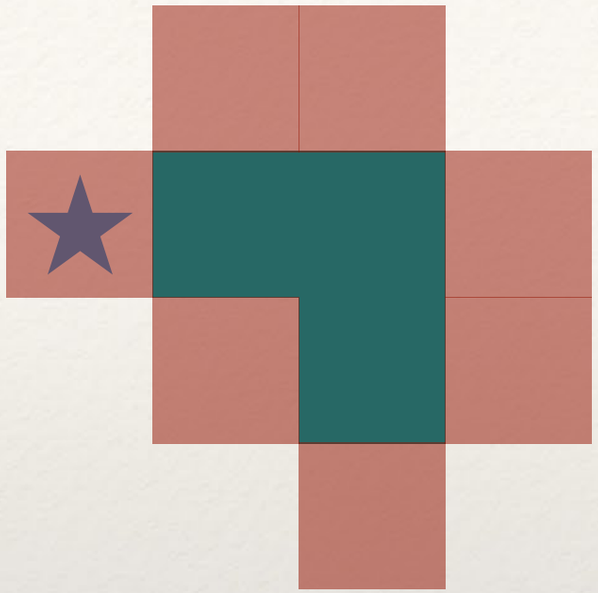
Boundary Description

Chain Codes

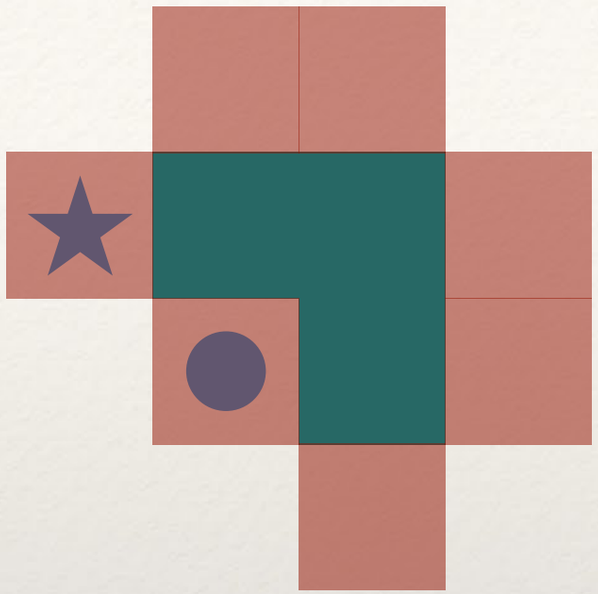
- Simple way of encoding a boundary.
  - Walk around the boundary and encode the direction you take on each step as a number
  - Then cyclically shift the code so it forms the smallest possible integer value (making it invariant to the starting point)



Chain Codes Example

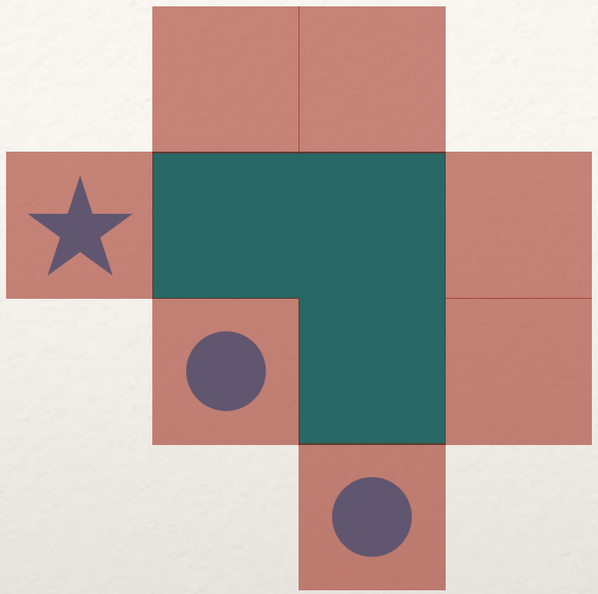


Chain Codes Example



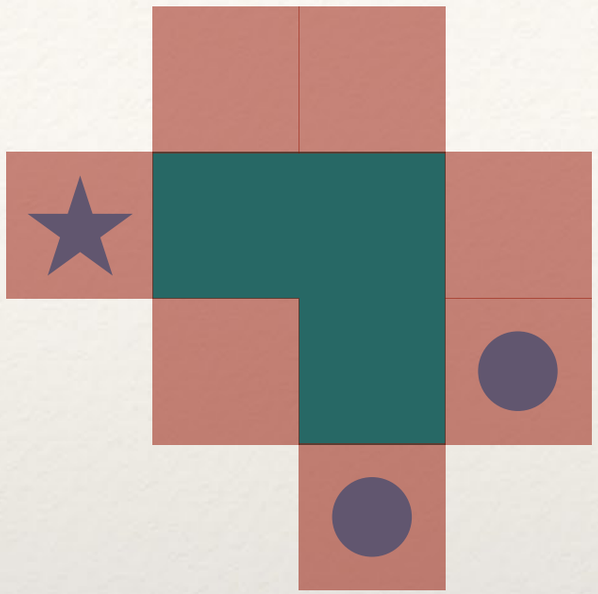
7

Chain Codes Example



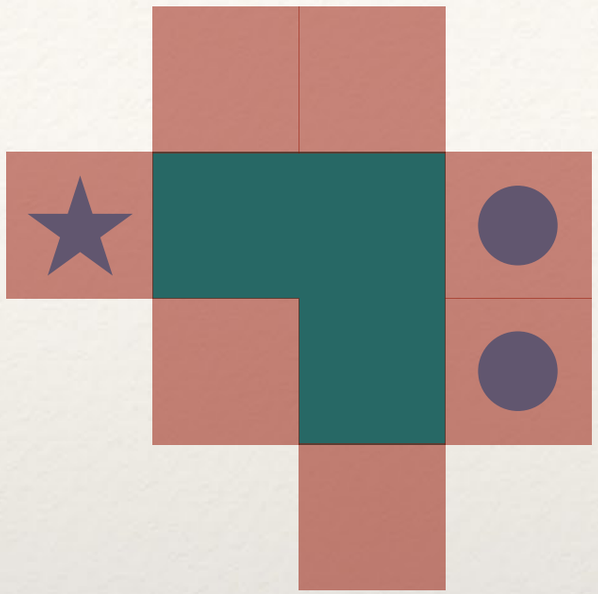
77

Chain Codes Example



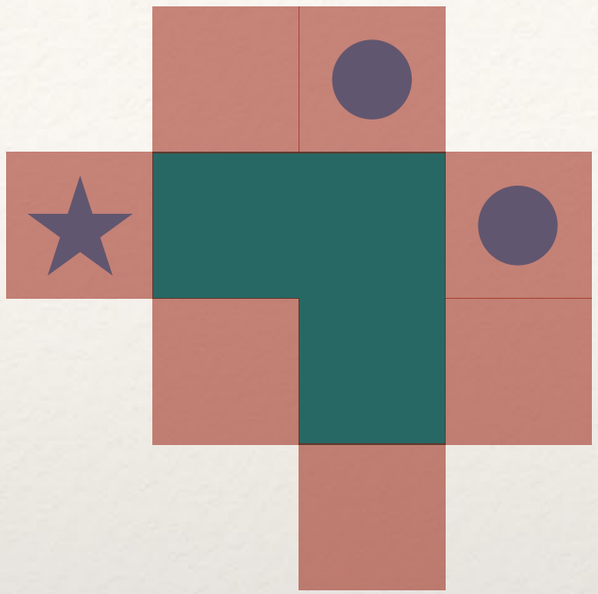
771

Chain Codes Example



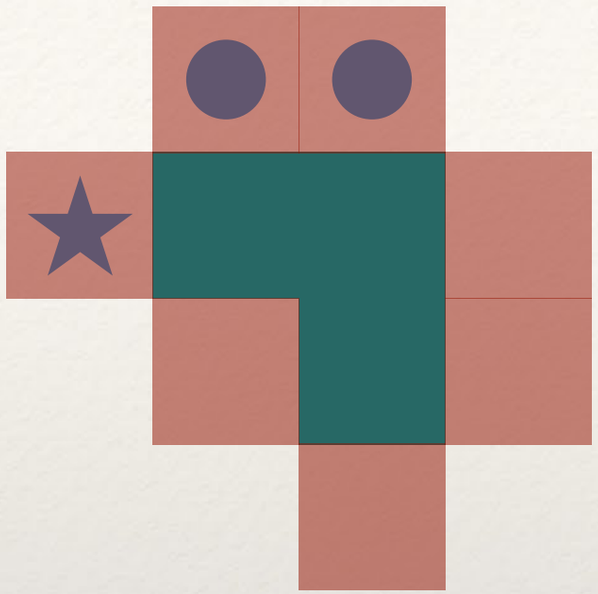
7712

Chain Codes Example



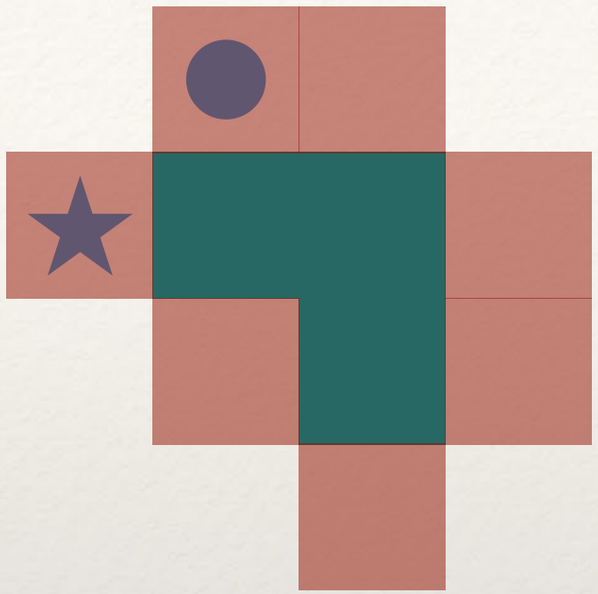
77123

Chain Codes Example



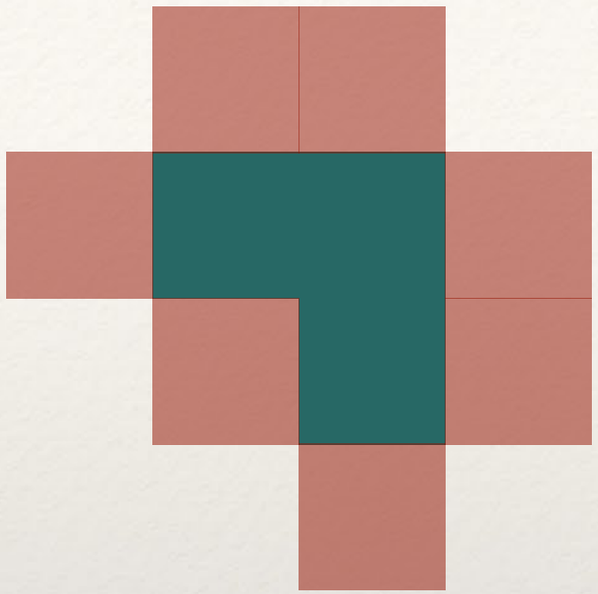
771234

Chain Codes Example



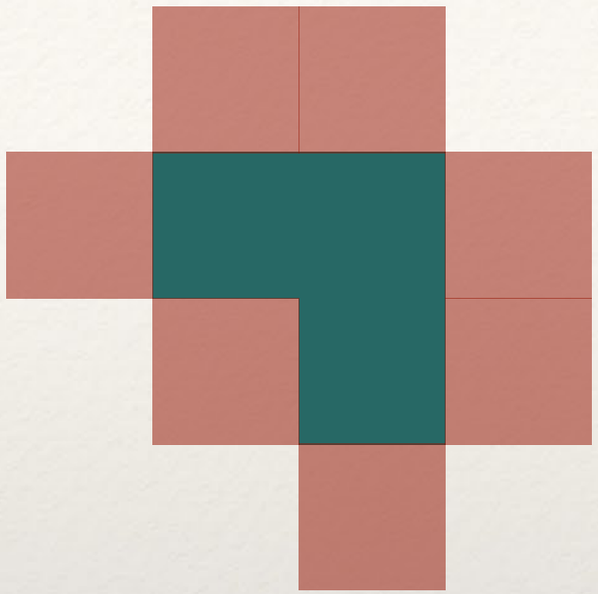
7712345

Chain Codes Example



7123457

Chain Codes Example



1234577

Chain Code Invariance

- Can be made rotation invariant:
  - Encode the differences in direction rather than absolute values.
- Can be made scale invariant:
  - Resample the component to a fixed size
    - Doesn't work well in practice…

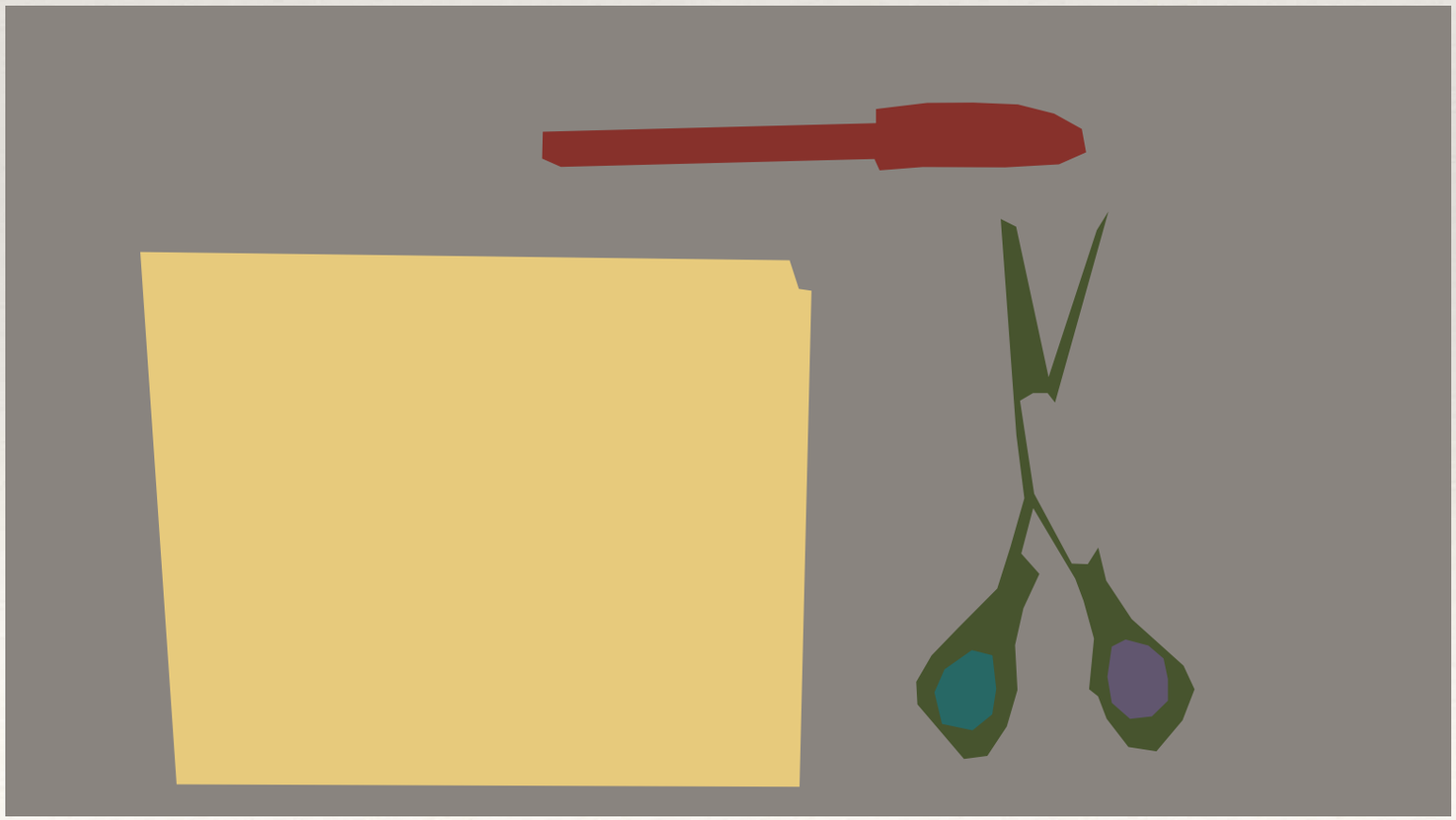
Chain Code Advantages and Limitations

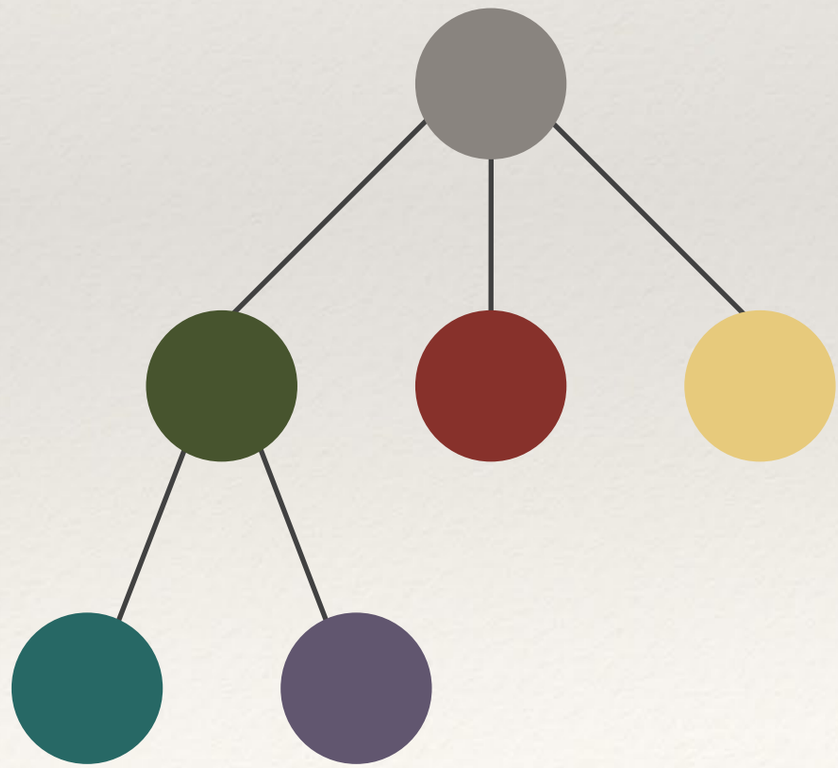
- Can be used for computing perimeter area, moments, etc.
  - Perimeter for an 8-connected chain code is N(even numbers in code) + $\sqrt{2}$N(odd numbers in code)
- Practically speaking, not so good for shape matching
  - Problems with noise, resampling effects, etc
  - Difficult to find good similarity/distance measures

Describing Multiple Shapes

Region Adjacency Graphs

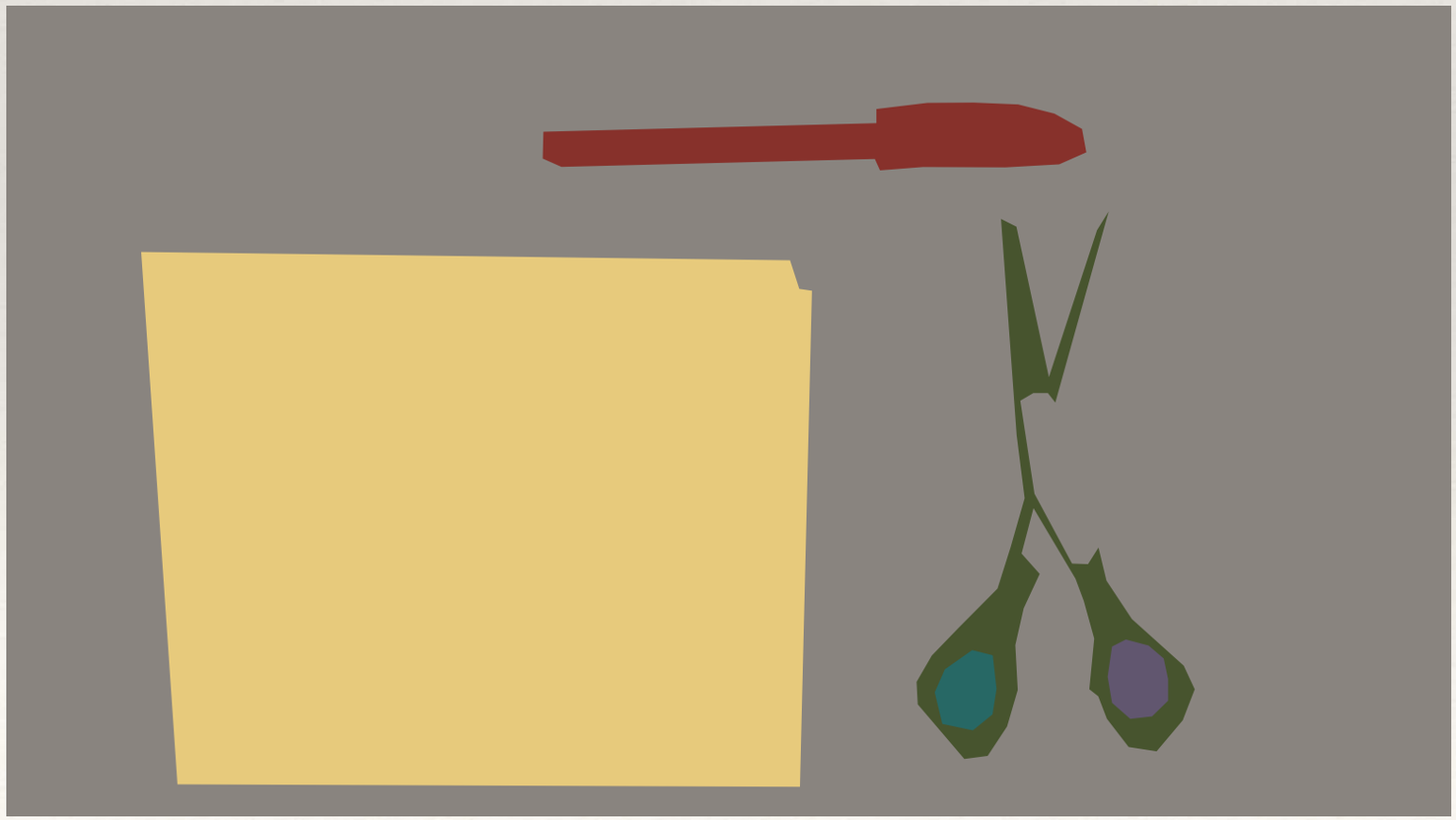
- Build a graph from a set of connected components
  - Each node corresponds to a component
  - Nodes connected if they share a border

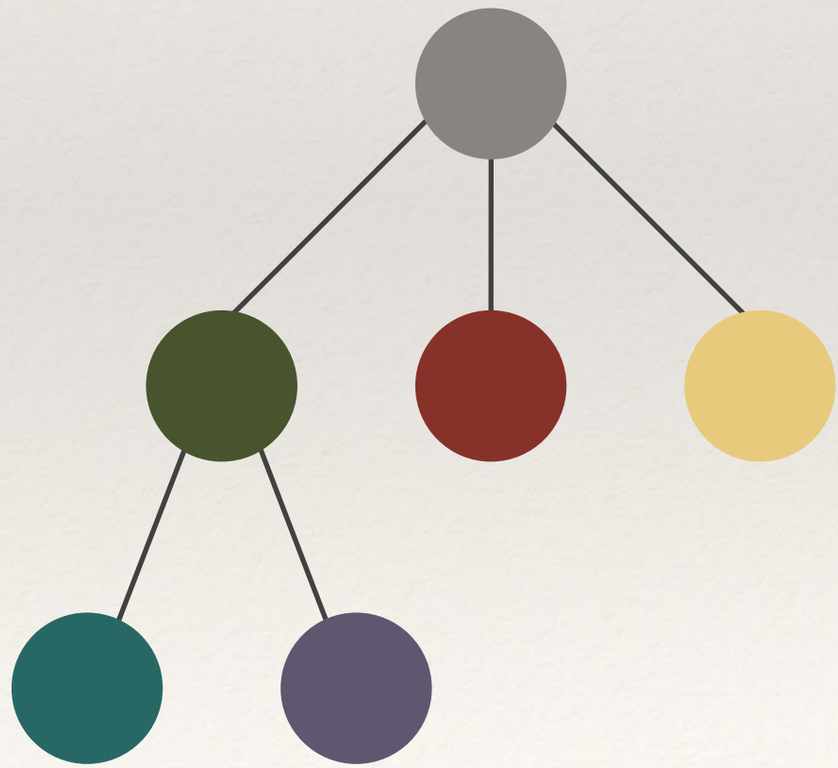




Region Adjacency Graphs

- Can easily detect patterns in the graph
  - e.g. “a node with one child with four children”
- Invariant to non-linear distortions, but not to *occlusion*



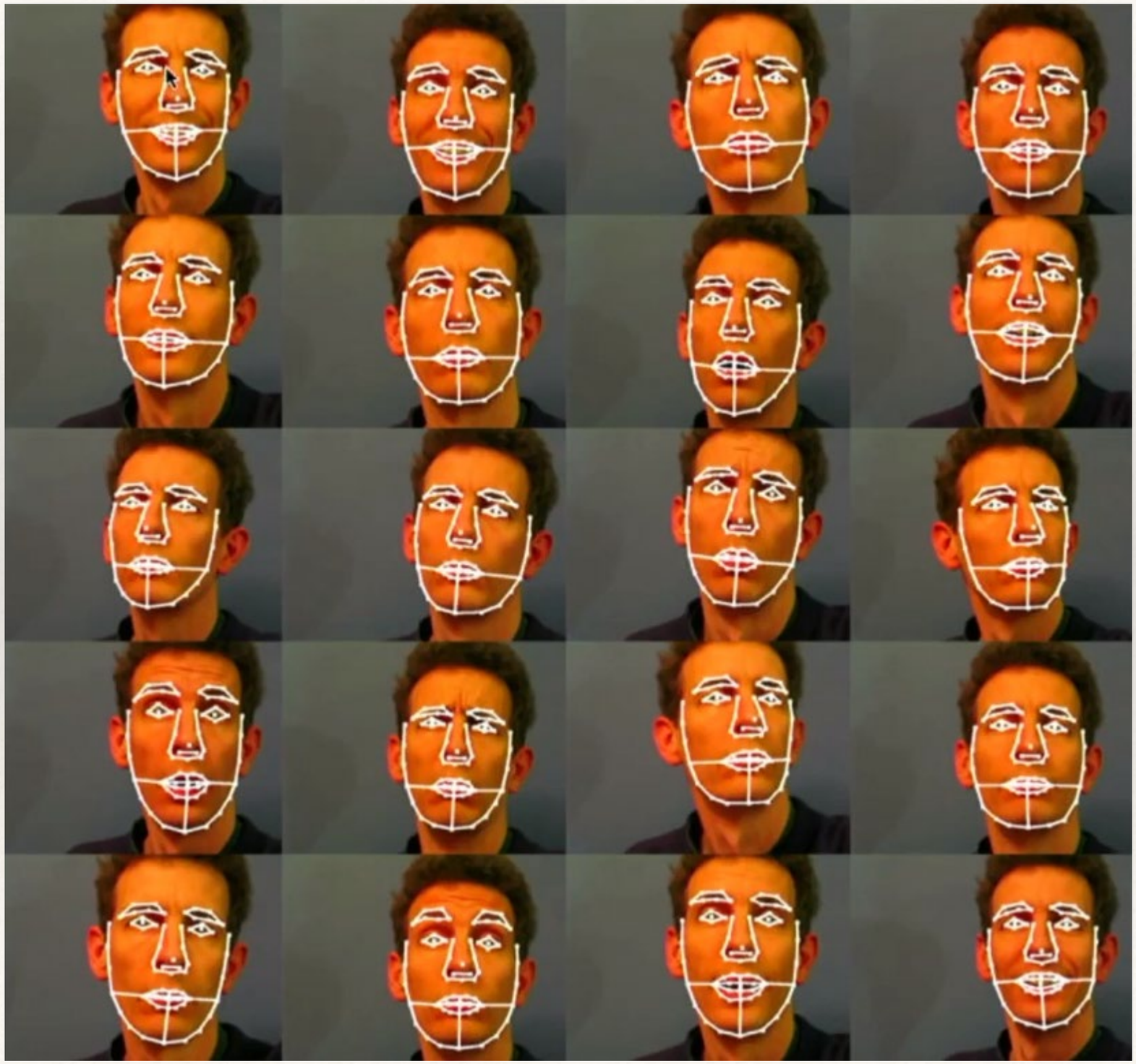


Flexible Shape Modelling and Fitting

Point Distribution Models (PDM)

- Idea: learn a low dimensional parametric model of how the shape of an object can vary.
  - Shape represented by a fixed number of 2D points at distinguishable locations on the object.
  - e.g. for faces, points at the corners of the mouth and eyes, tip of the noise, etc
  - Need a training set of images with the same points marked…

Point Distribution Models



Procrustes Analysis

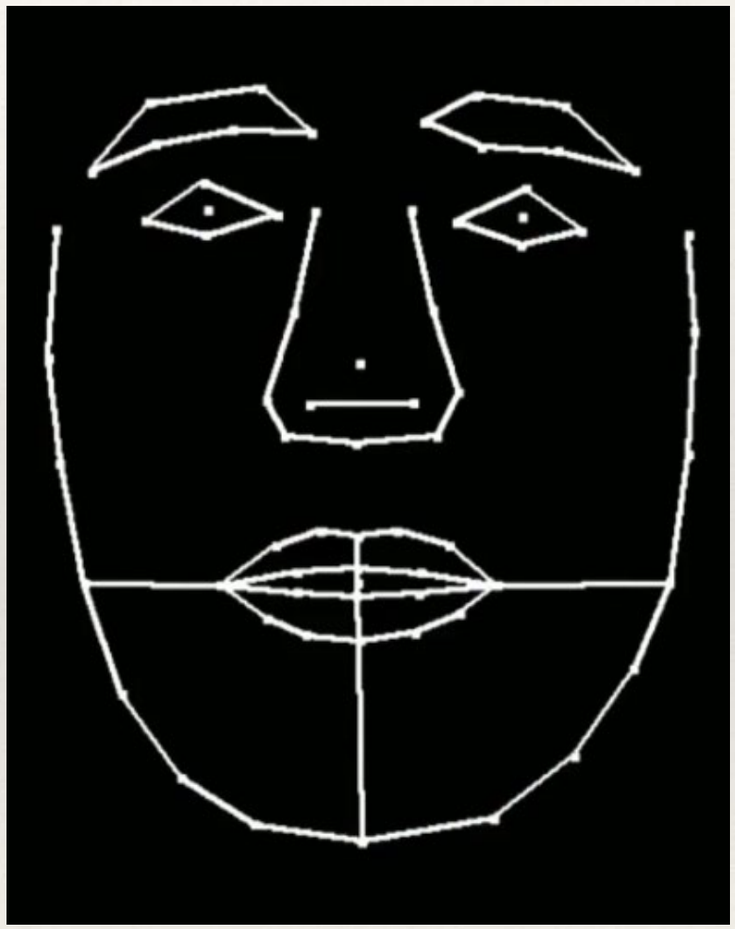
- Need to align (rotate and scale) all training points.
  - All point-sets must have the same size and be about the origin.
  - *Generalised procrustes analysis* is an iterative technique for achieving this:

1. arbitrarily choose a reference shape (typically by selecting it among the available instances and centering it about the origin)
2. superimpose all instances to current reference shape (by computing the optimal translation, scaling and rotation for each)
3. compute the mean shape of the current set of superimposed shapes
4. if the Procrustes distance (the Euclidean distance between [x1,y1,x2,y2,…] and [x’1,y’1,x’2,y’2,…]) between mean and reference shape is above a threshold, set reference to mean shape and continue to step 2.

Procrustes Analysis

Unaligned
(Original)

Aligned



Mean shape

Form the data points into a shape matrix

$$S = \begin{bmatrix} x_{11} & y_{11} & x_{12} & y_{12} & \dots & \dots & x_{1M} & y_{1M} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_{N1} & y_{N1} & x_{N2} & y_{N2} & \dots & \dots & x_{NM} & y_{NM} \end{bmatrix}$$

Each row corresponds to an image (it's essentially a featurevector for the shape in that image!)

Each pair of columns contains (x, y) coordinates of aligned corresponding points across the images

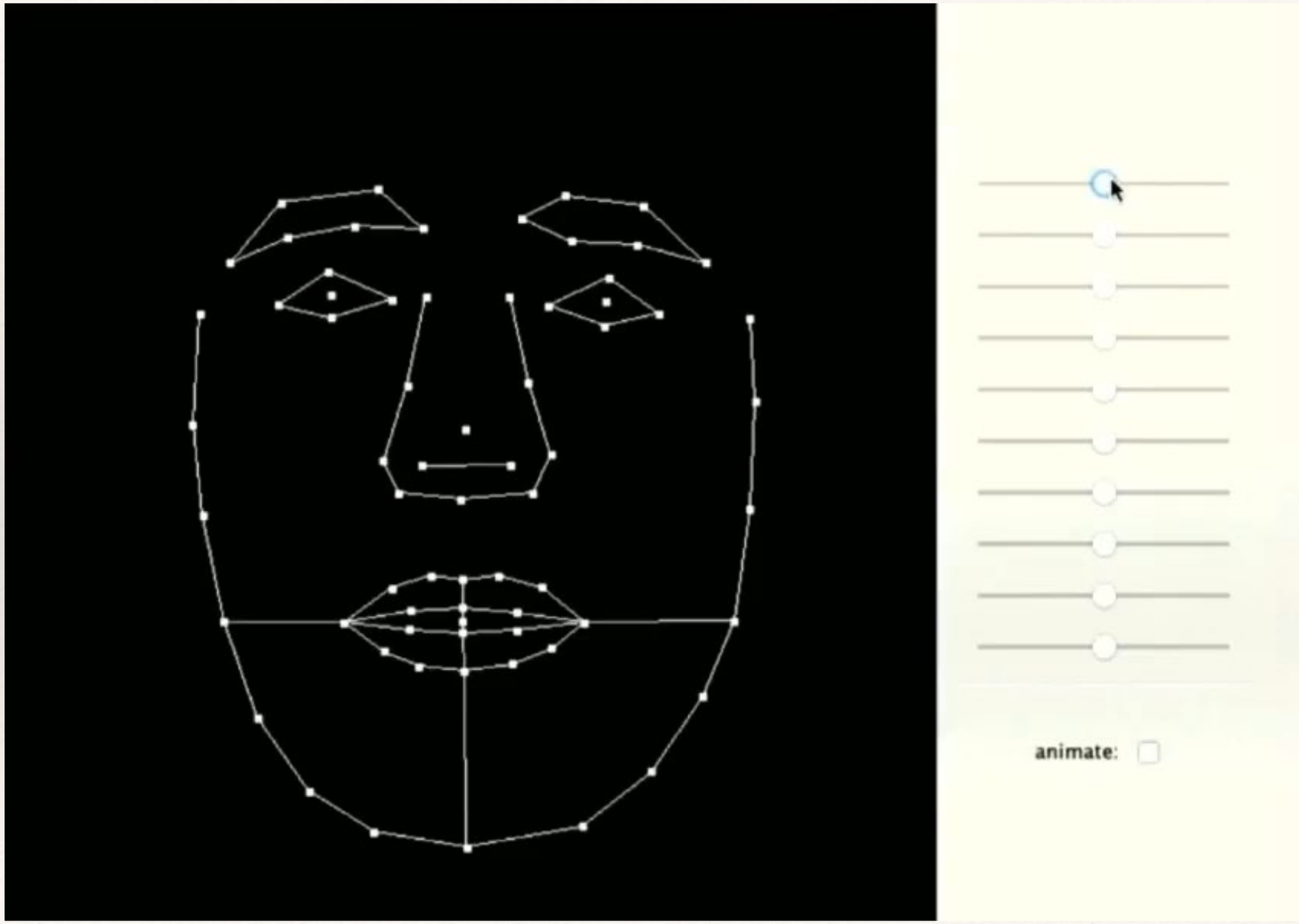
Apply PCA to learn a low-dimensional basis

$$\mathbf{S}^{\mathrm{T}}\mathbf{S} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\mathrm{T}}$$

Q can be used to *generate* shapes from a low-dimensional weight vector, w:

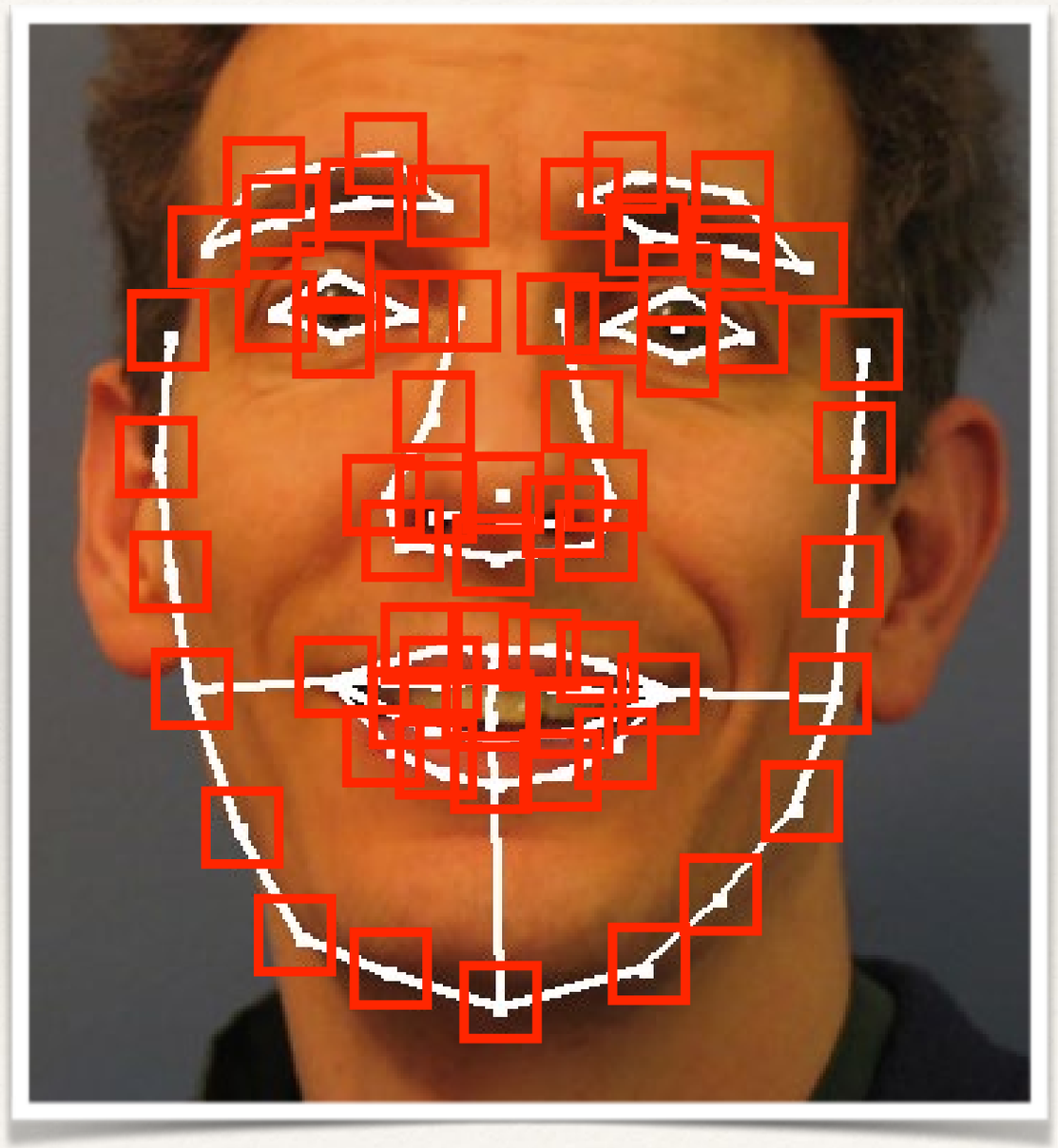
$$v = w\mathbf{Q}^{\mathrm{T}}$$

Demo: PDM



Active Shape Models and Constrained Local Models

- ASMs/CLMs extend a PDM by also learning local appearance around each point
  - Typically just as an image *template*
- Using a constrained optimisation algorithm, the shape can be optimally fitted to an image
  - Constraints:
    - *Plausible shape*
    - *Good template matching*



CLM Tracker

https://vimeo.com/75659453

Active Appearance Models

- AAMs go even further and model the global appearance of the shape
  - …using Eigenfaces!
- Optimiser fits the model to new images by searching for both plausible shape and plausible global appearance

Face2Face

J. Thies et al. "Face2Face: Real-time Face Capture and Reenactment of RGB Videos" CVPR 2016

Summary

- Many different ways to describe shape of connected components
  - Choice really depends on required invariance
- Multiple shapes can efficiently be represented by a RAG
- Point distribution models apply PCA to x-y coordinate pairs across multiple images to produce a low-dimensional parametric model
  - ASMs/CLMs also model local appearance, and can use this to optimise the fit of the model parameters to match an image

Further reading and exercises

- **Further reading**
  - Mark's book (3rd edition), covers shape description in Chapter 7 (including detailed information on Fourier descriptors).
  - Sonka, Hlavac and Boyle "Image Processing, Analysis and Machine Vision" (chapter 6 in the second edition).
  - CLM Tracker https://www.auduno.com/2014/01/05/fitting-faces/
- **Practical exercises**
  - org.openimaj.image.connectedcomponent.proc package for some more advanced shape features
  - Connected Components with OpenCV
    - https://www.pyimagesearch.com/2021/02/22/opencv-connected-component-labeling-and-analysis/
    - https://docs.opencv.org/4.5.5/de/d01/samples_2cpp_2connected_components_8cpp-example.html
  - CLM Tracker: org.openimaj.image.processing.face.tracking.clm.
  - CLMFaceTracker: https://github.com/auduno/clmtrackr

* *Acknowledgements: B*ased on earlier Computer Vision lecture slides by Dr. Jon Hare.