2008 SIAM CONFERENCE ON DATA MINING

APRIL 24-26, 2008

HYATT REGENCY HOTEL ATLANTA, GEORGIA

Important note: *These slides undergo constant revisions, visit www.cs.ucr.edu/~eamonn/ for the latest version*

Fair Use Agreement

This agreement covers the use of all slides, please read carefully.

You may freely use these slides for teaching, if

- You send me an email telling me the class number/ university in advance.
- My name and email address appears on the first slide (if you are using all or most of the slides), or on each slide (if you are just taking a few slides).

- You may freely use these slides for a conference presentation, if
  - You send me an email telling me the conference name in advance.
  - My name appears on each slide you use.

- You may not use these slides for tutorials, or in a published work (tech report/ conference paper/ thesis/ journal etc). If you wish to do this, email me first, it is highly likely I will grant you permission.

Mining Massive Collections of Shapes and Time Series: With Case Studies in Anthropology and Astronomy

SDM 2008 Tutorial

Eamonn Keogh, UCR
eamonn@cs.ucr.edu

Outline of Tutorial I

- Introduction, Motivation
- The ubiquity of time series and shape data
- Examples of problems in time series and shape data mining
- The utility of distance measurements
- Properties of distance measures
    - Euclidean distance
    - Dynamic time warping
    - Longest common subsequence
- Why no other distance measures?
- Preprocessing the data
- Invariance to distortions
- Spatial Access Methods and the curse of dimensionality
- Generic dimensionality reduction
    - Discrete Fourier Transform
    - Discrete Wavelet Transform
    - Singular Value Decomposition
    - Adaptive Piecewise Constant Approximation
    - Piecewise Linear Approximation
    - Piecewise Aggregate Approximation

  Very Briefly
- Why Symbolic Approximation is different
- Why SAX is the best symbolic approximation

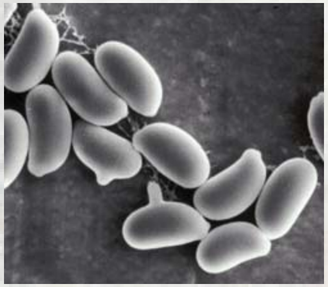
Outline of Tutorial II

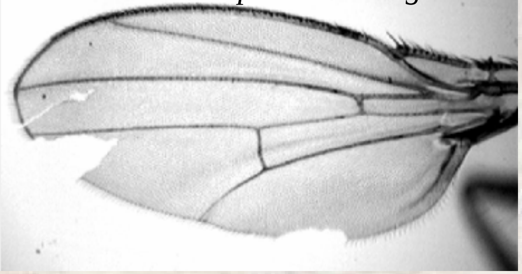
In both shape and time series, we consider:

- Novelty detection (finding unusual shapes or subsequences)
- Motif discovery (finding repeated shapes or subsequences)
- Clustering
- Classification
- Indexing
- Visualizing massive datasets
- Open problems to solve
- Summary, Conclusions

The Ubiquity of Shape

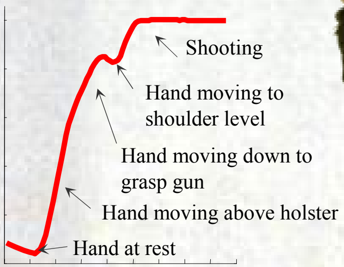
…butterflies, fish, petroglyphs, arrowheads, fruit fly wings, lizards, nematodes, yeast cells, faces, historical manuscripts…



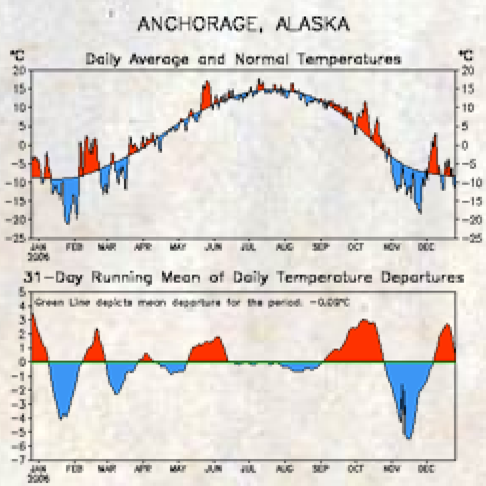


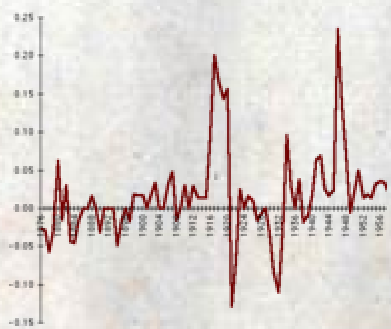
Drosophila melanogaster

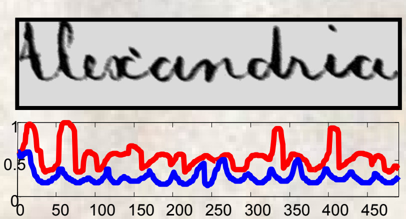
The Ubiquity of Time Series

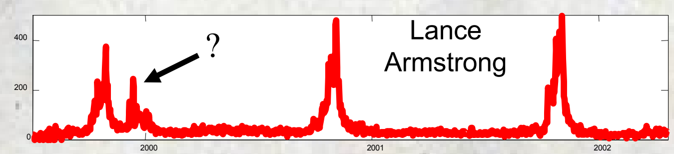


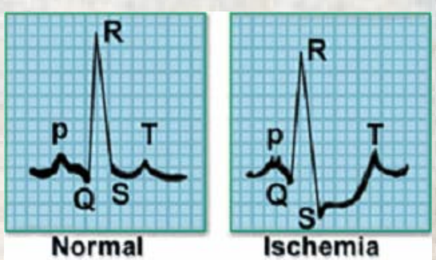
Don't Shoot! *Motion capture, meteorology, finance, handwriting, medicine, web logs, music…*

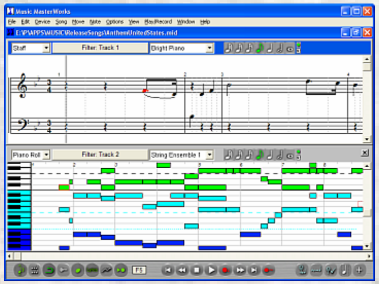








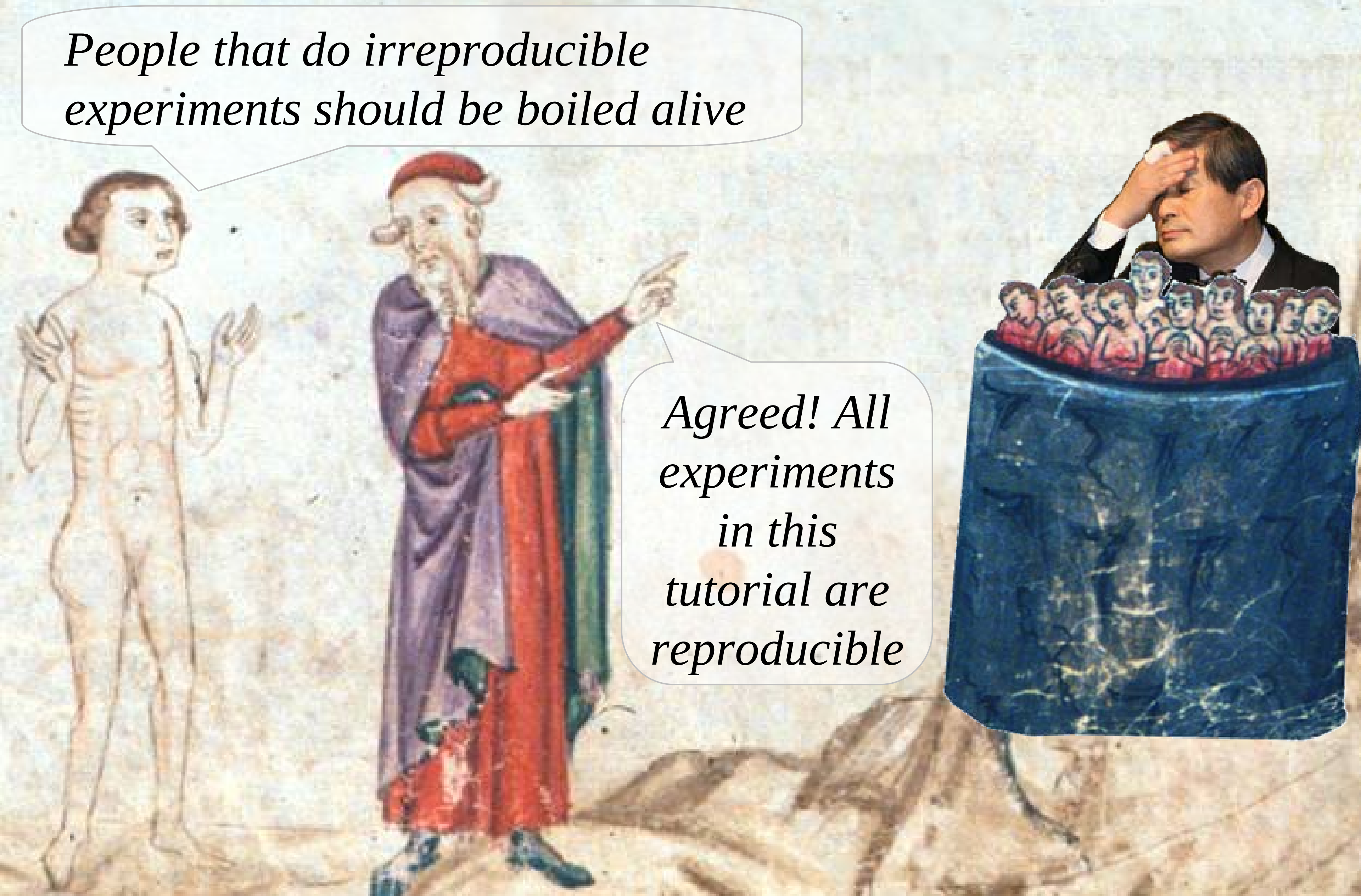




Examples of problems in time series and shape data mining

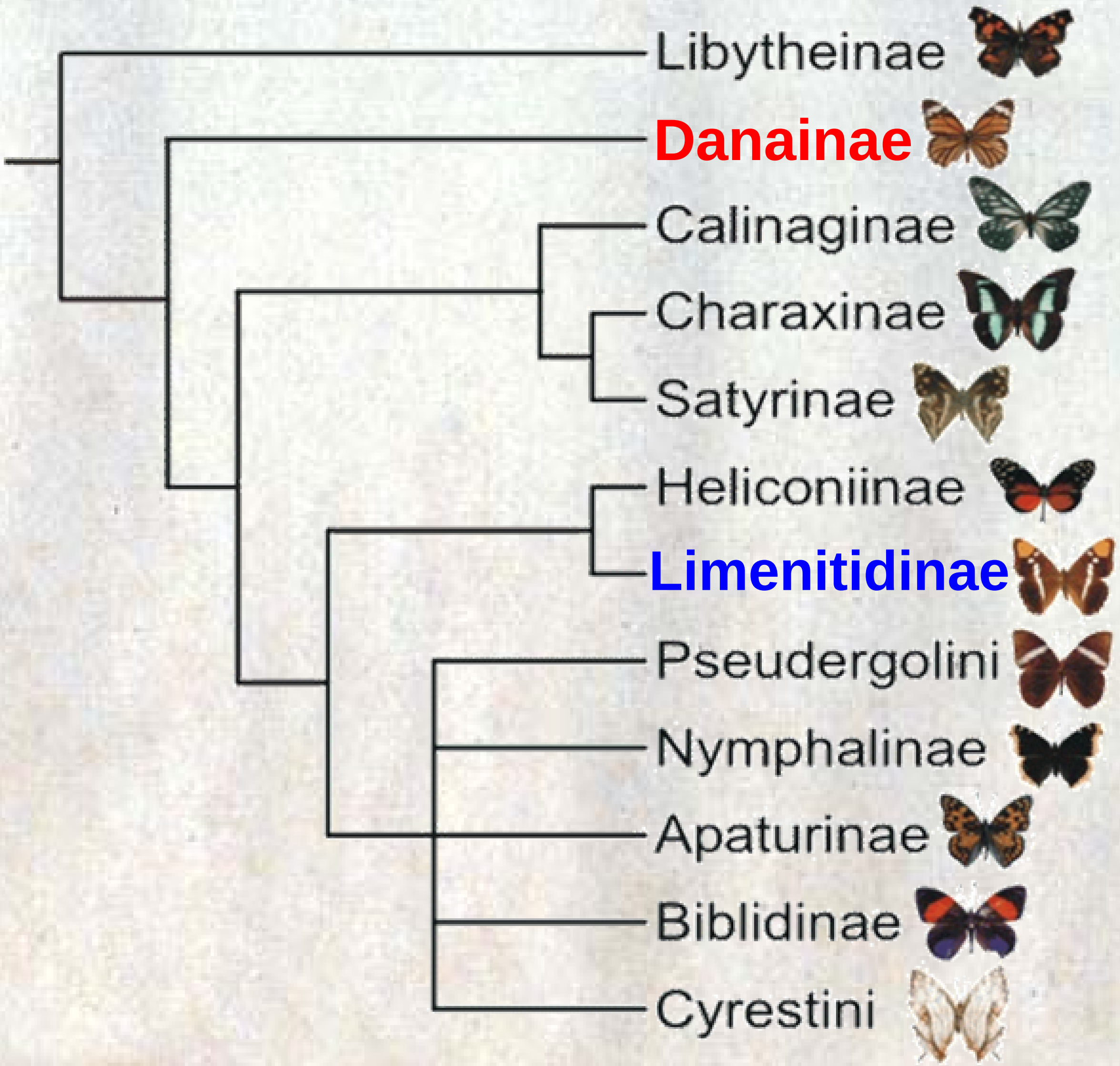
In the next few slides we will see examples of the kind of problems we would like to be able to solve, then later we will see the necessary tools to solve them

All our Experiments are Reproducible!

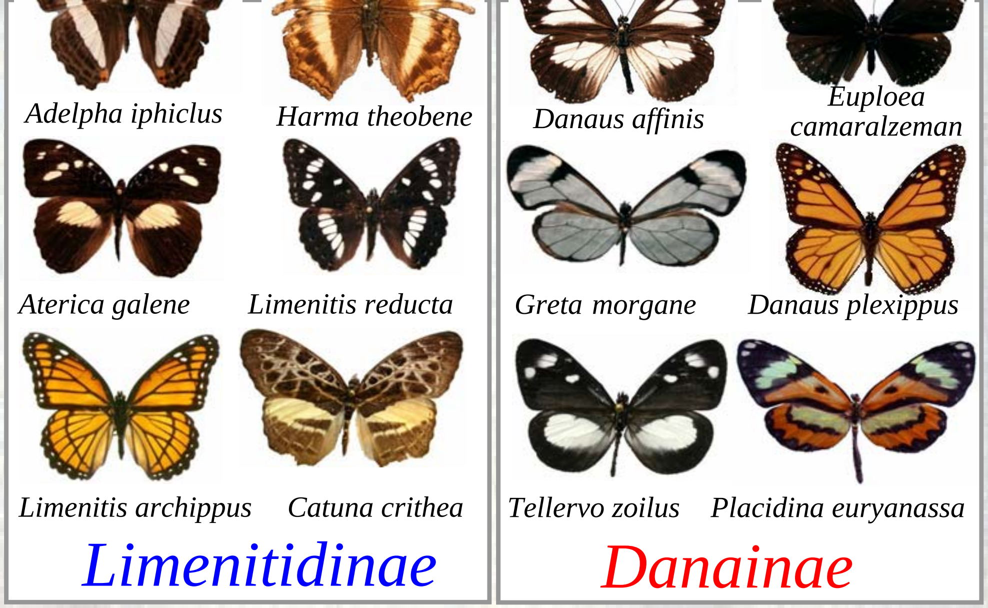


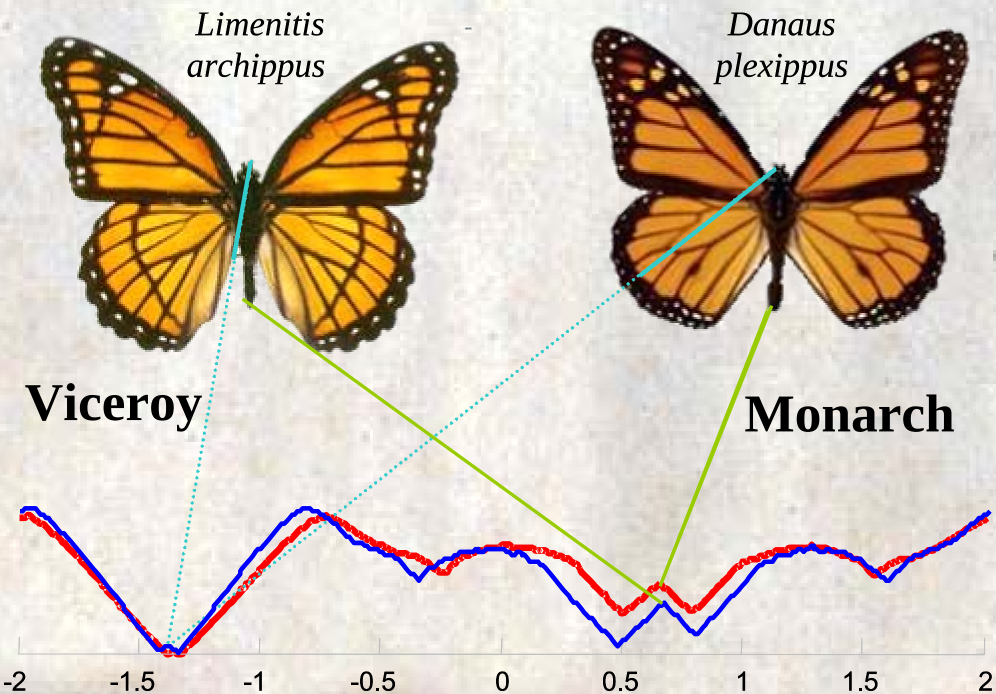
Example 1: Join

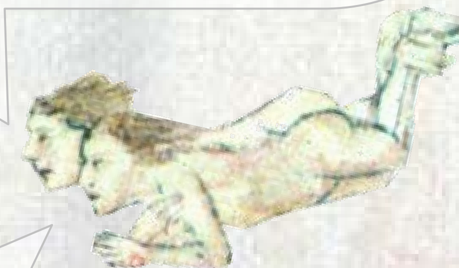
Given two data collections, link items occurring in each



We can take two different families of butterflies, Limenitidinae and Danainae, and find the most similar shape between them







*Why would the two most similar shapes also have similar colors and patterns? That **can't** be a coincidence. This is an example of Müllerian mimicry*

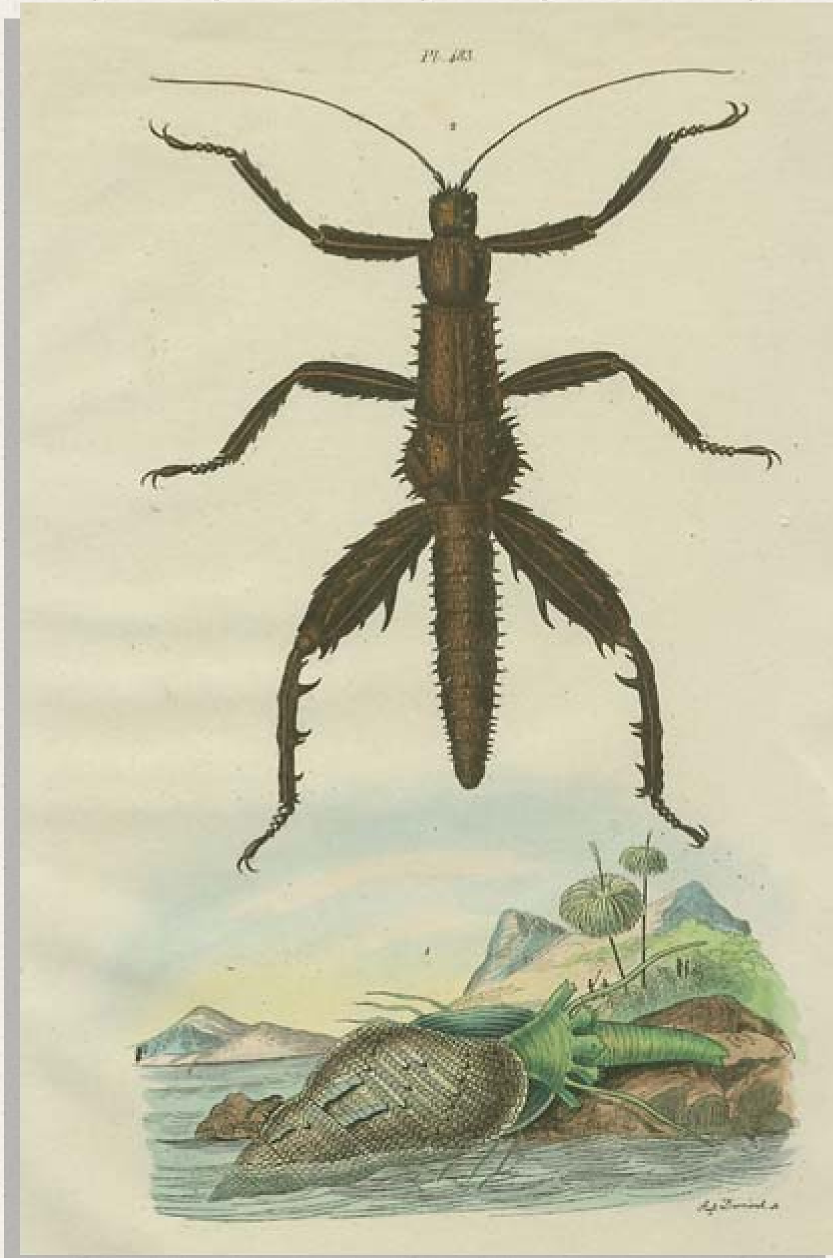
Not Batesian mimicry as commonly believed

*.. so similar in coloration that I will put them both to one*****


Photo by Lincoln Brower

Example 2: Annotation

Given an object of interest, automatically obtain additional information about it.



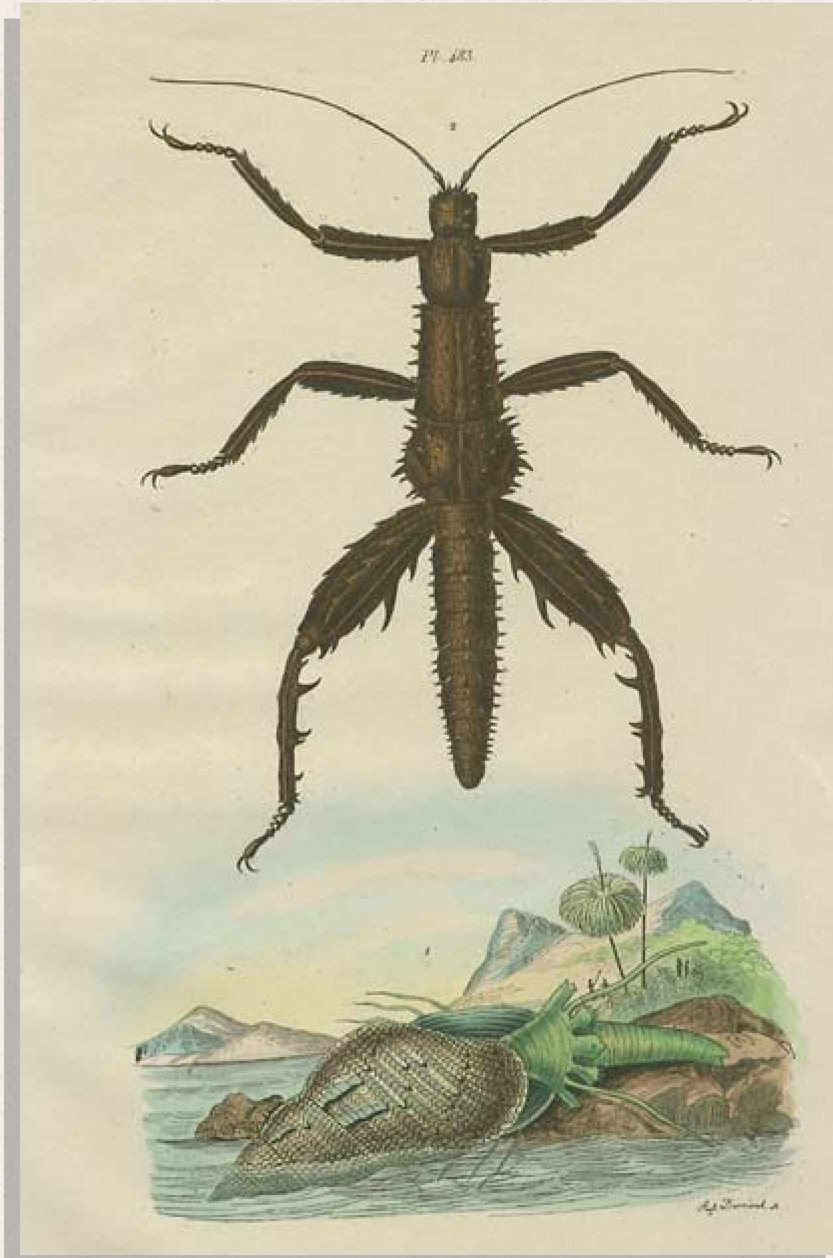
Friedrich Bertuch's *Bilderbuch fur Kinder* (Weimar, 1798–1830)

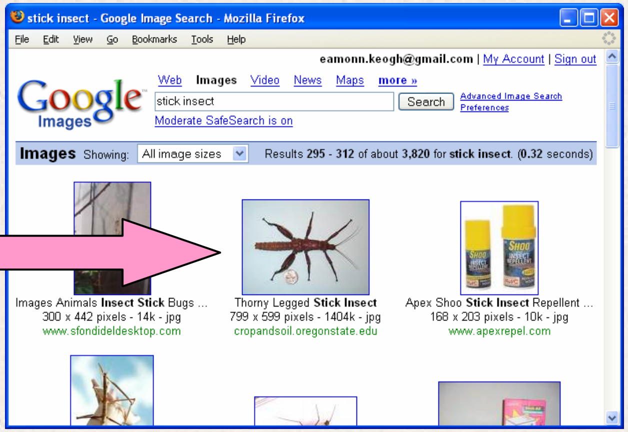
This page was published in 1821

Bilderbuch is a children's encyclopedia of natural history, published in 237 parts over nearly 40 years in Germany.

Suppose we encountered this page and wanted to know more about the insect. The back of the page says "*Stockinsekt* " which we might be able to parse to "*Stick Insect*", but what kind? How large is it? Where do they live?

Suppose we issue a query to Google search for "*Stick Insect*" and further filter the results by shape similarity….

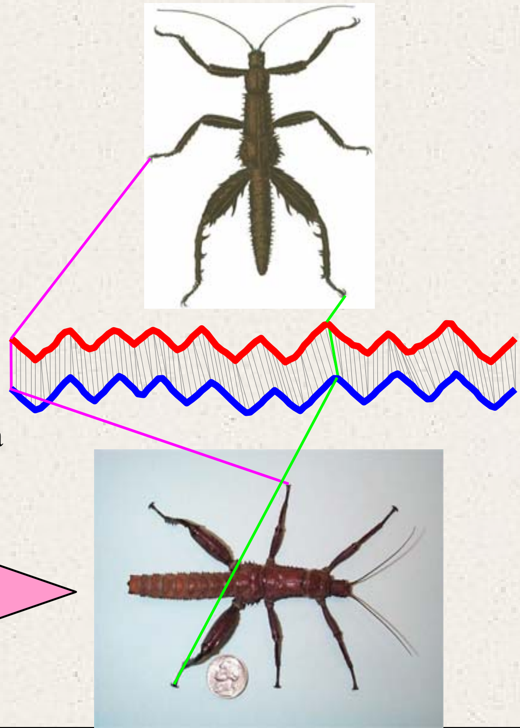




Most images returned by the Google image query "stick insect" do not segment into simple shapes, but some do, including the 296th one.

It looks like our insect is a Thorny Legged Stick Insect, or *Eurycantha calcarata* from Southeast Asia.

Note that in addition to rotation invariance our distance measure must be invariant to other differences. The real insect has a tail that extends past his legs, and asymmetric positions of limbs etc.



Example 3: Query by Content

Given a large data collection, find the *k* most similar objects to an object of interest.

Petroglyphs

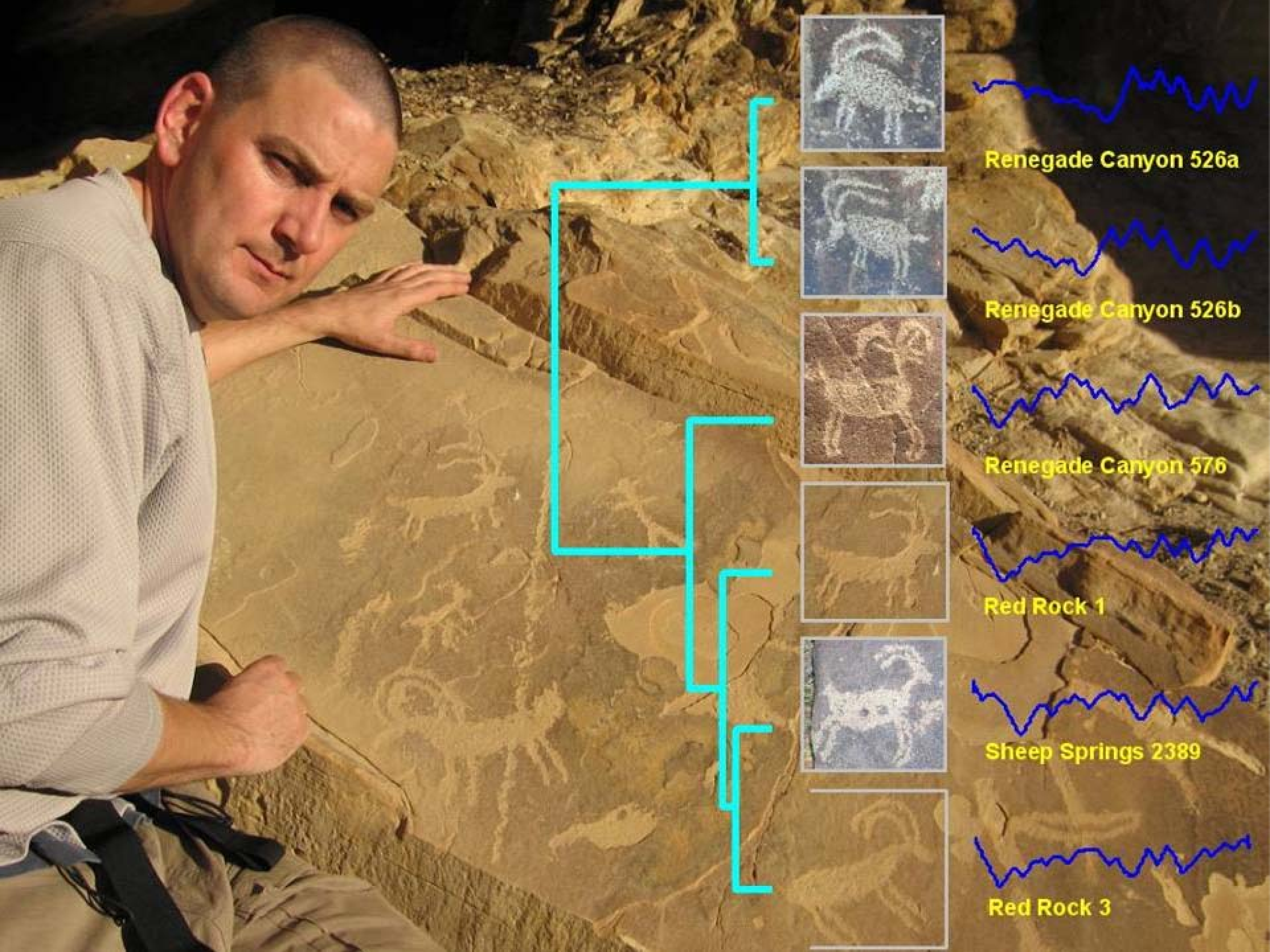
- They appear worldwide
- Over a million in America alone
- Surprisingly little known about them

Petroglyphs are images incised in rock, usually by prehistoric peoples. They were an important form of pre-writing symbols, used in communication from approximately 10,000 B.C.E. to modern times. **Wikipedia**

*who so sketched out the shapes there?**

*.. they would strike the subtlest minds with awe**

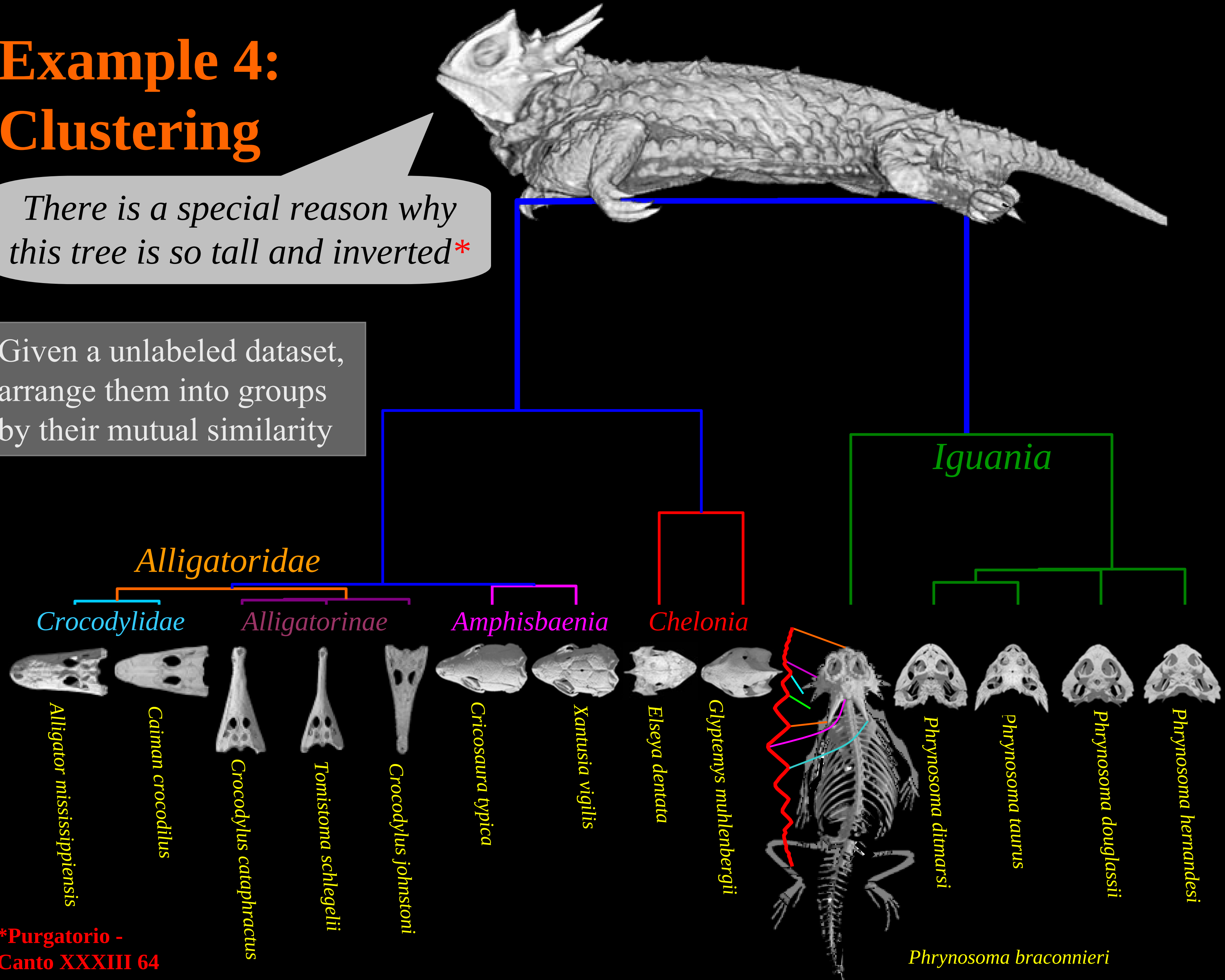
***Purgatorio -- Canto XII 6**


Renegade Canyon 526a
Renegade Canyon 526b
Renegade Canyon 576
Red Rock 1
Sheep Springs 2389
Red Rock 3

Example 4: Clustering

*There is a special reason why this tree is so tall and inverted**

Given a unlabeled dataset, arrange them into groups by their mutual similarity

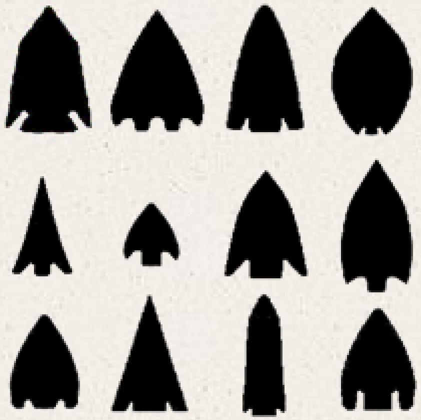


***Purgatorio - Canto XXXIII 64**

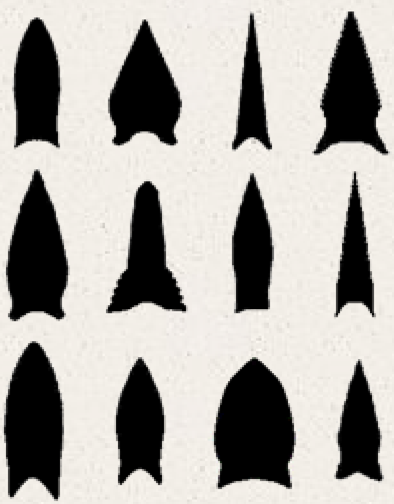
Example 5: Classification

Given a labeled training set,
classify future *unlabeled* examples

Basal



Articulate



What type of arrowhead is this?

*For he is well placed among the fools who does not distinguish one class from another**

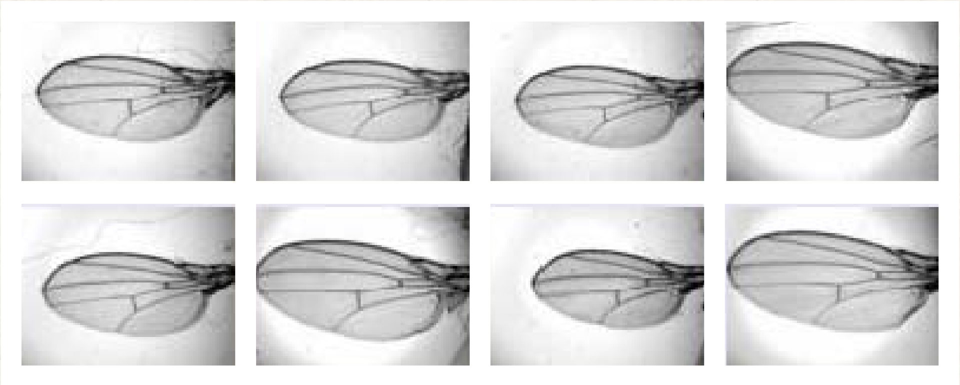
*Paradiso -- Canto XIII 115

Example 6: Anomaly Detection (*Discords*)

*…you are merely like imperfect insects**

Given a large collection of objects, find the one that is most different to all the rest.

A subset of 32,028 images of Drosophila wings



***Purgatorio -- Canto X 127**

Example 7: Repeated Pattern Discovery (*Motifs*)

Given a large collection of objects, find the pair that is most similar.

each one is alike
in size and
*rounded shape**

Blythe, California

Baker California

***Inferno -- Canto XIX 15**

Example(s) 8: Human Motion

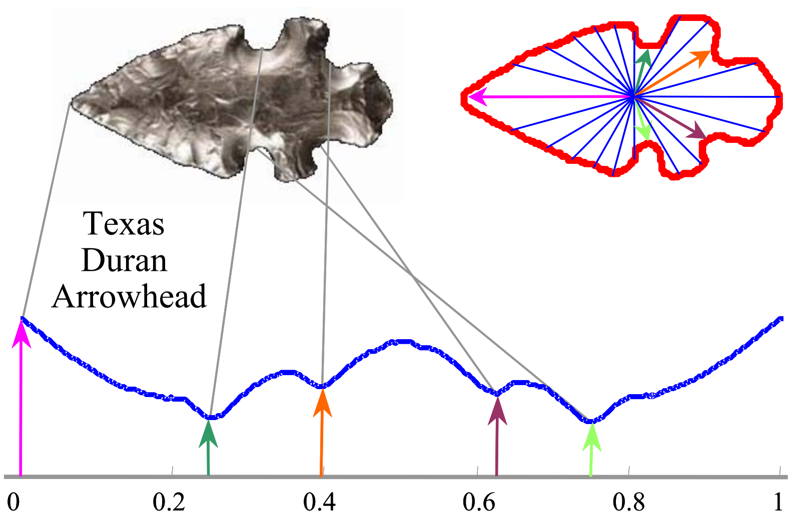
- **Join**
- **Annotation**
- **Query-by-Content**
- **Clustering**
- **Classification**
- **Anomaly Detection**
- **Motif Discovery**

***Inferno -- Canto VI**

MoCap Image by Meredith & Maddock

Two Kinds of Shape Matching

"rigid"

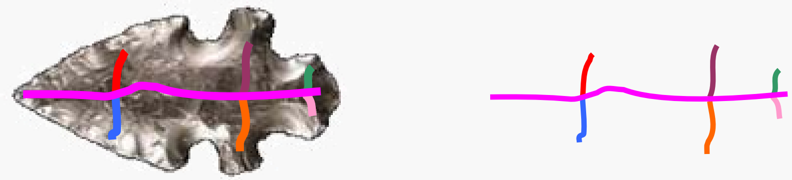


Convert shape to pseudo time series or feature vector. Use time series distance measures or vector distance measures to measure similarity.

*We **only** consider this approach in this tutorial.*

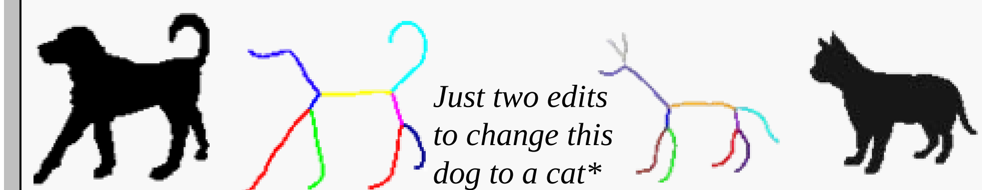
It works well for the butterflies, fish, petroglyphs, arrowheads, fruit fly wings, lizards, nematodes, yeast cells, faces, historical manuscripts etc discussed at the beginning of this tutorial.

"flexible"



Key Ideas: *Convert shape to graph/tree*

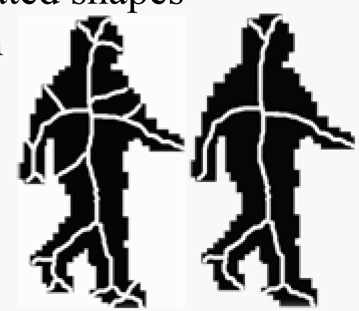
Use graph/tree edit distance to measure similarity



• Some shapes are already "graph like"

• Needed for articulated shapes

• The shape to graph transformation is very tricky#

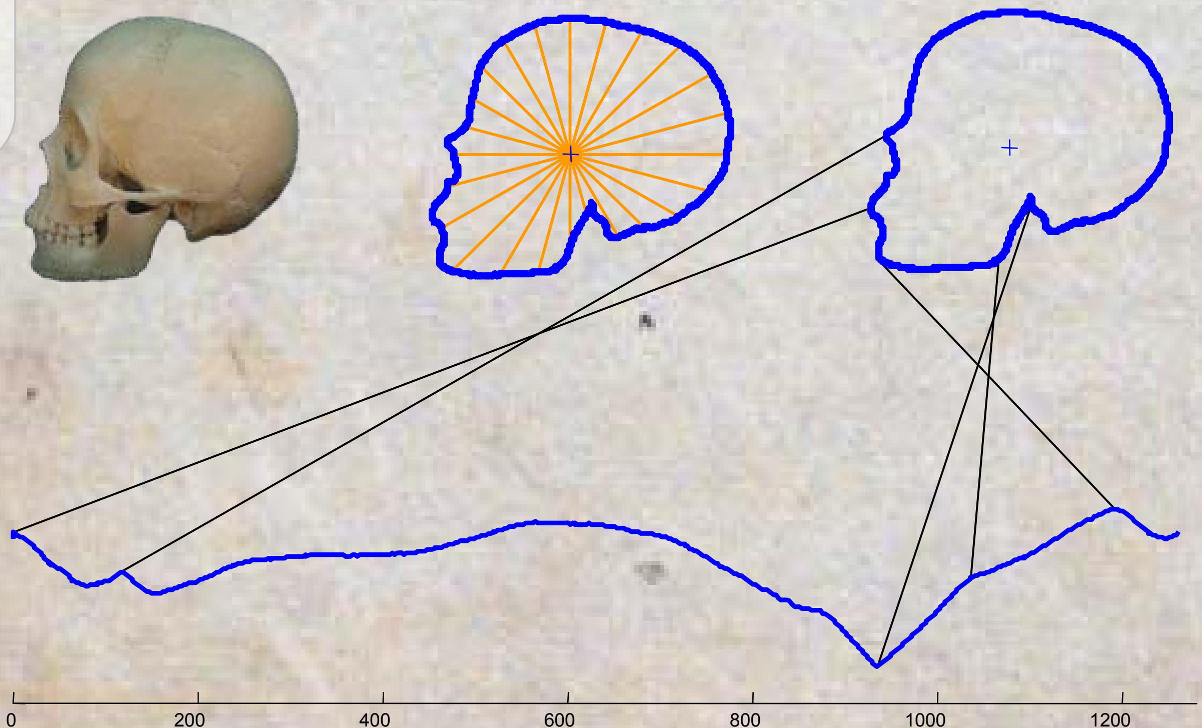


We do not further discuss these ideas, see "shock graph" work of Sebastian, Klein and Kimia* and the work of Latecki# and others

We can convert shapes into a 1D signal. Thus can we remove information about *scale* and *offset*.

Rotation we must deal with in our algorithms…

*…it seemed to change its shape, from running lengthwise to revolving round…**

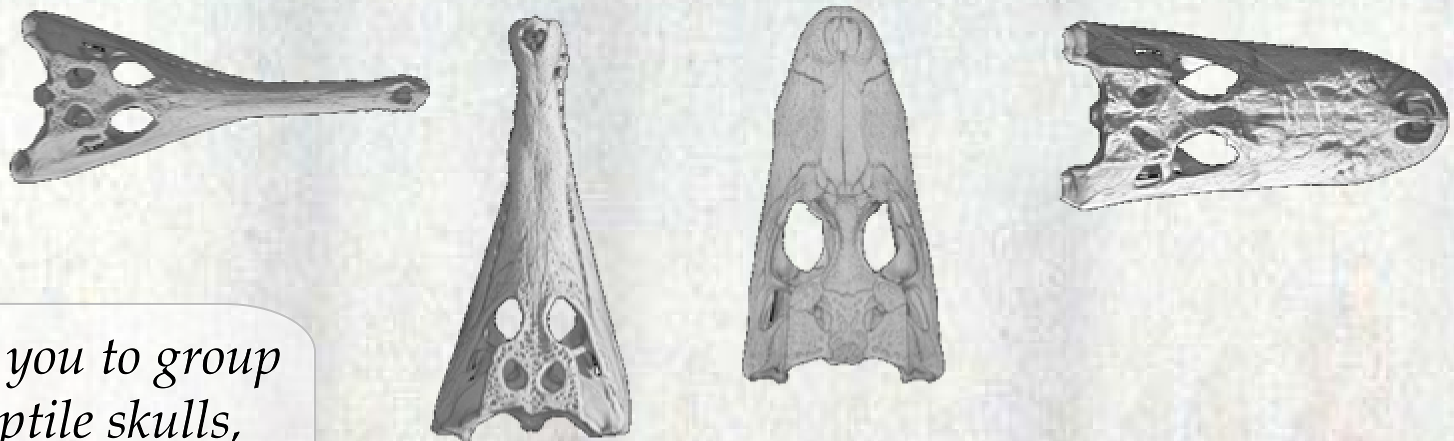


There are many other 1D representations of shape, and the algorithms shown in this tutorial can work with *any* of them

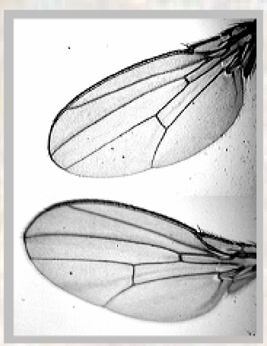
*Paradiso -- Canto XXX, 90.

Shape Representations

*For virtually all shape matching problems, rotation is **the** problem*



If I asked you to group these reptile skulls, rotation would not confuse you



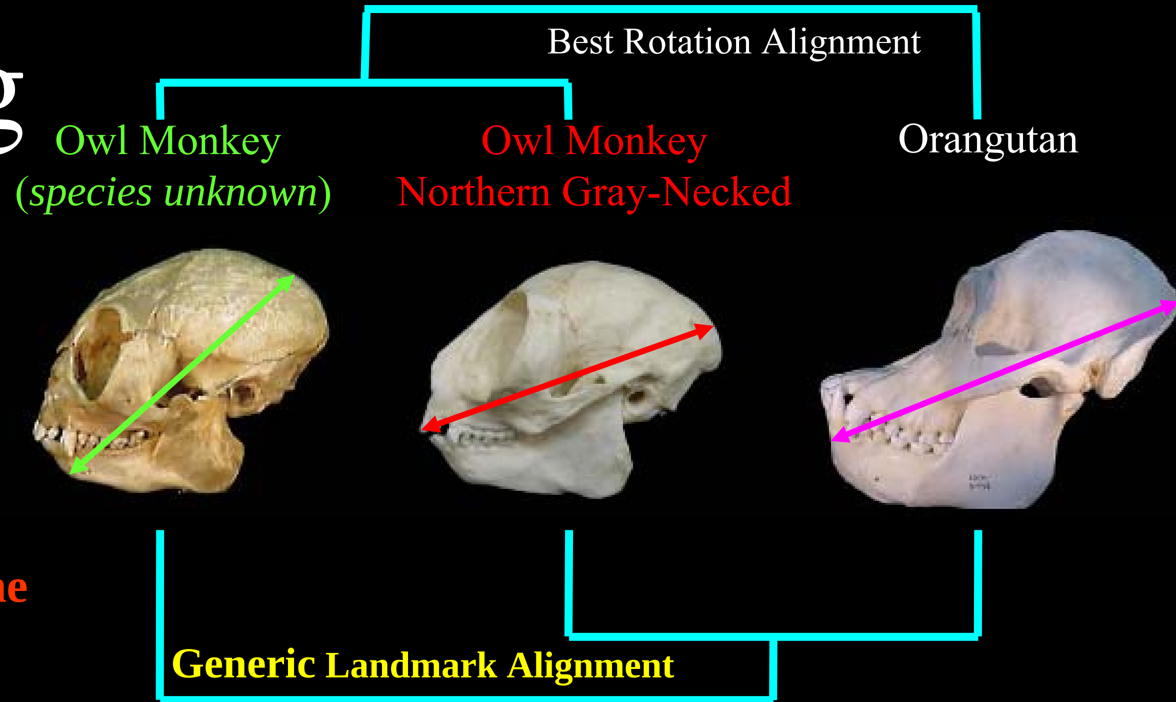
There are two ways to be rotation invariant

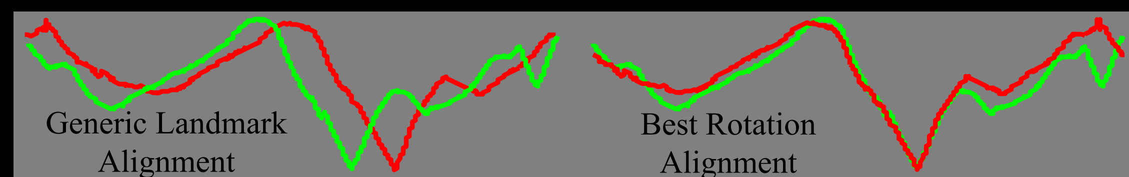
1) Landmarking: Find the one “true” rotation
2) Rotation invariant features

Landmarking

- **Generic Landmarking**
Find the major axis of the shape and use that as the canonical alignment

- **Domain Specific Landmarking**
Find some fixed point in your domain, eg. the nose on a face, the stem of leaf, the tail of a fish …





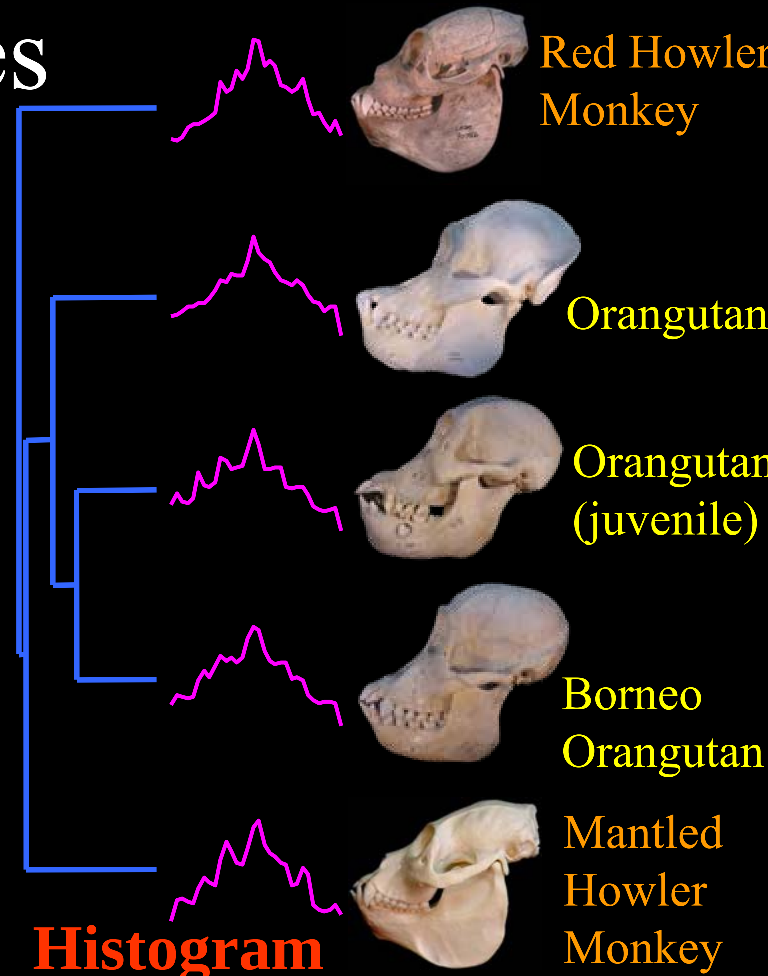
Domain Specific Landmarking

Rotation invariant features

Possibilities include:

Ratio of perimeter to area, fractal measures, elongatedness, circularity, min/max/mean curvature, entropy, perimeter of convex hull, **aspect ratio** and **histograms**

The problem with rotation invariant features is that in throwing away rotation information, you must invariably throw away useful information



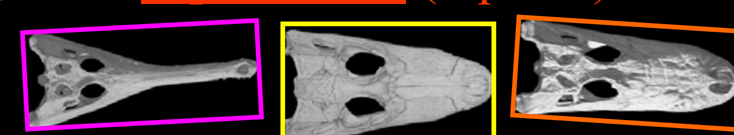
aspect ratio (monkeys)

0.73	0.49	0.47

works here

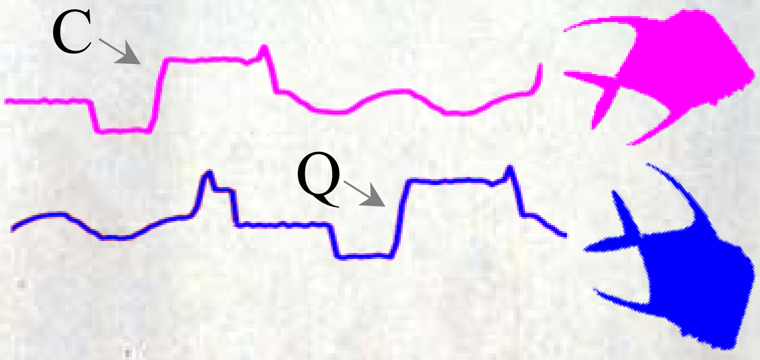
not here

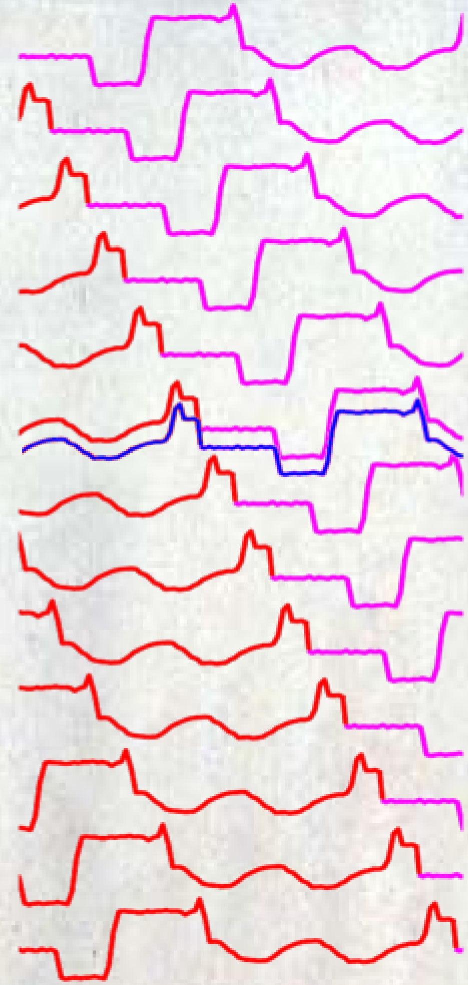
aspect ratio (reptiles)



0.41	0.54	0.43

The easy way to achieve rotation invariance is to hold one time series C *fixed, and compare it to every circular shift of the other time series, which is represented by the matrix* ***C***





algorithm: [dist] = Test_All_Rotations(Q,***C***)
dist = *infinty*
for *j = 1* **to** *n*
TempDistance = *Some_Dist_Function*(Q, ***C****j*)
if TempDistance < dist
dist = TempDistance;
end;
end;
return[dist]

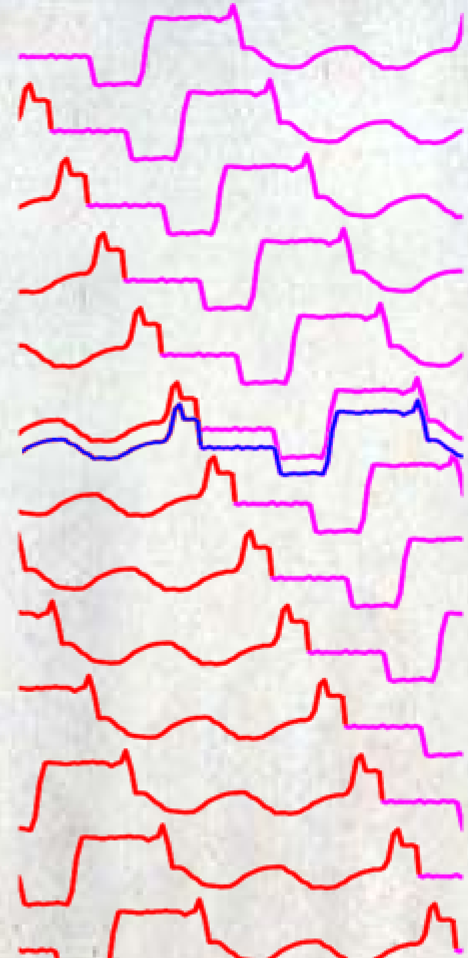
It sucks being a grad student

$$\boldsymbol{C} = \begin{Bmatrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{Bmatrix}$$

The strategy of testing all possible rotations is very very slow

People have suggested various tricks for speedup, like only testing 1 in 5 of the rotations

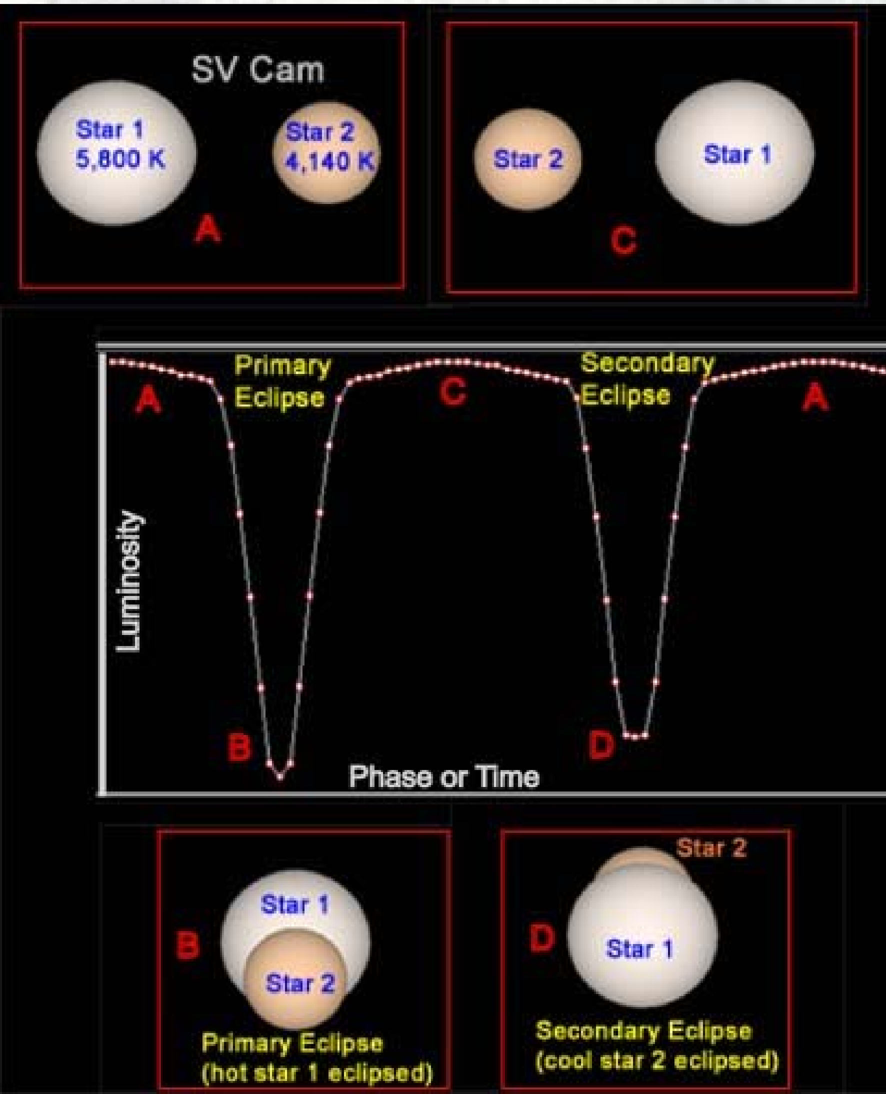
However there now exists a simple ***exact*** ultrafast, indexable way to do this*



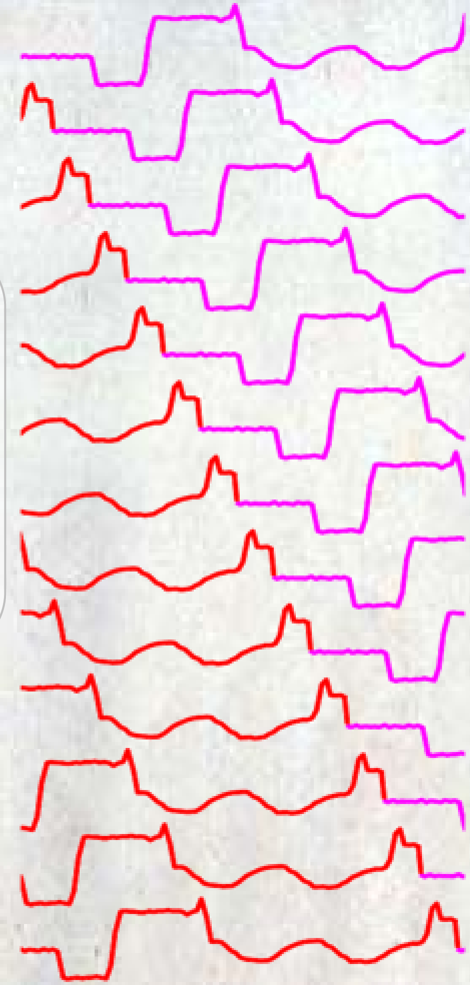
$$\mathbf{C} = \begin{Bmatrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{Bmatrix}$$

***VLDB06:** LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures.

The need for rotation invariance shows up in real time series, as in these Star Light Curves



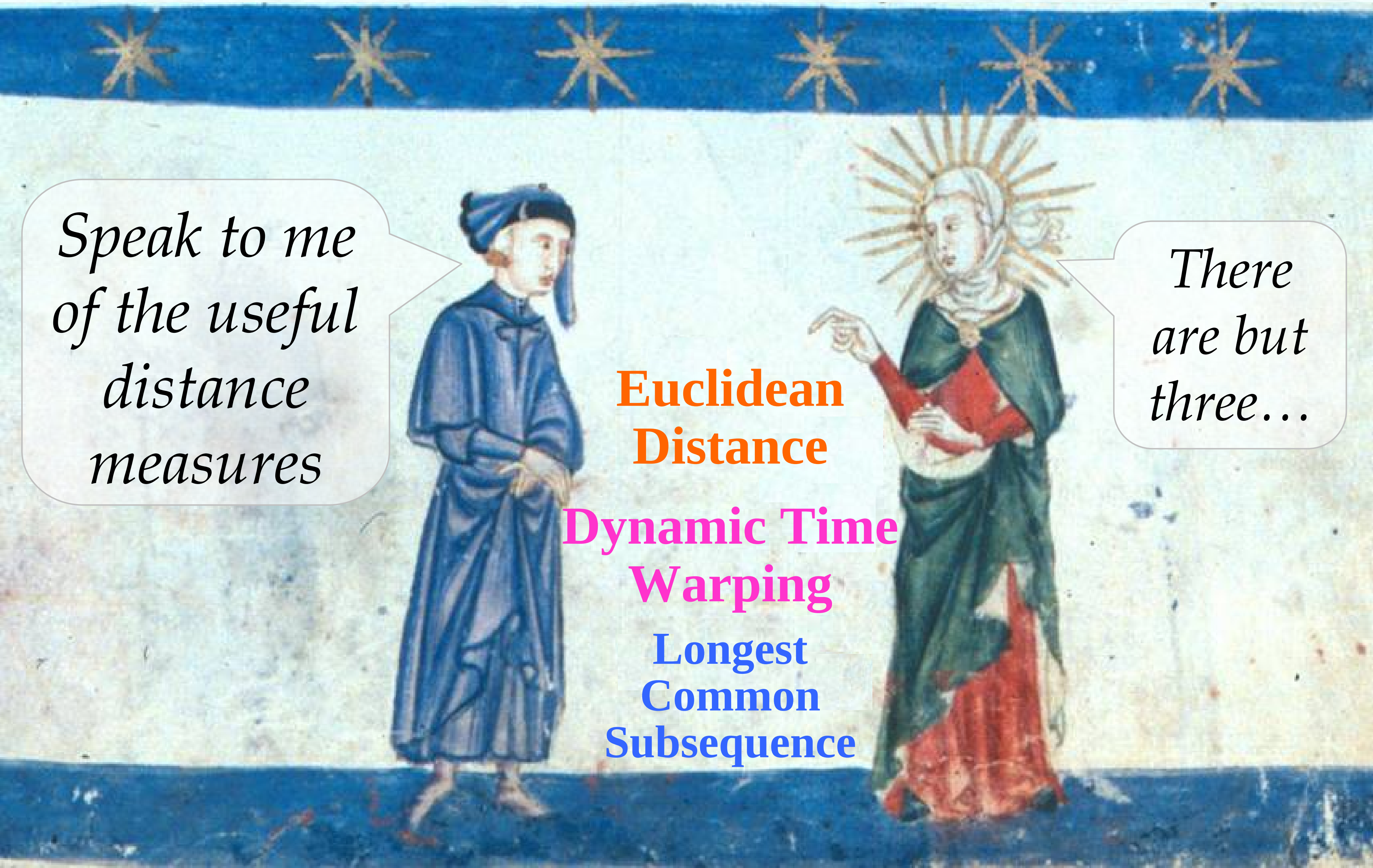
*I saw above a million burning lamps,
A Sun kindled every one of them, as our sun lights the stars we glimpse on high**

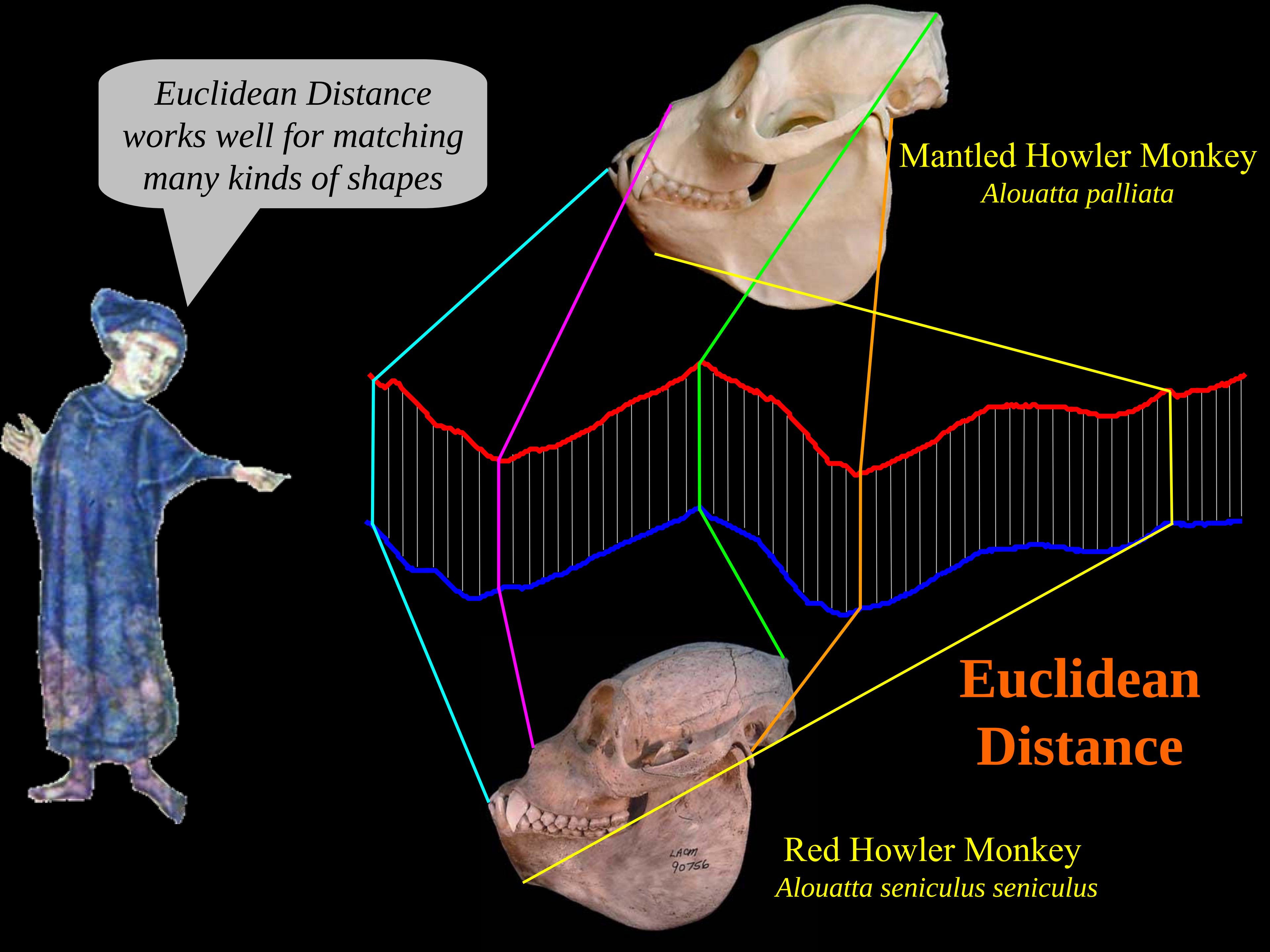


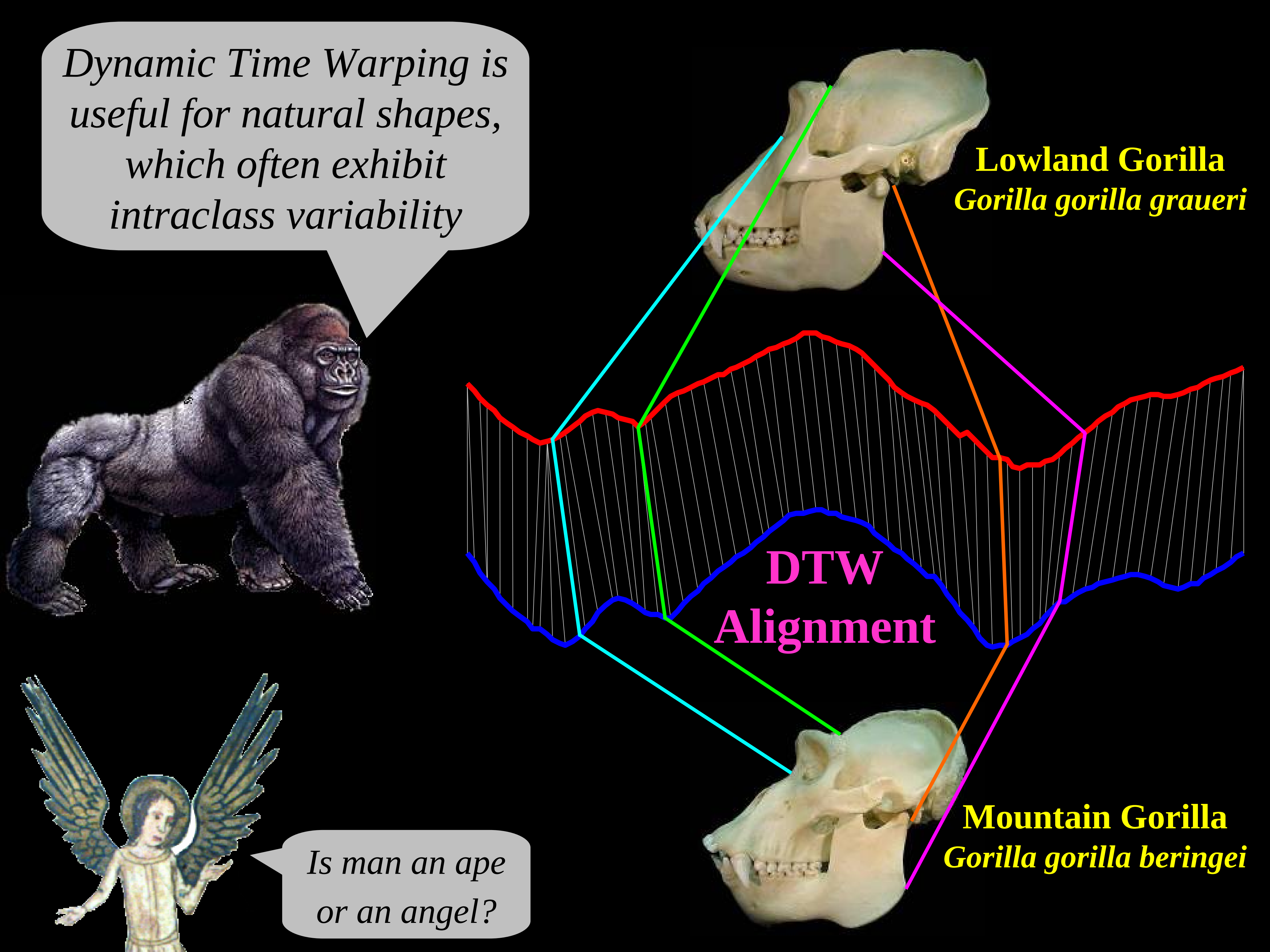
$$\boldsymbol{C} = \left\{ \begin{matrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{matrix} \right\}$$

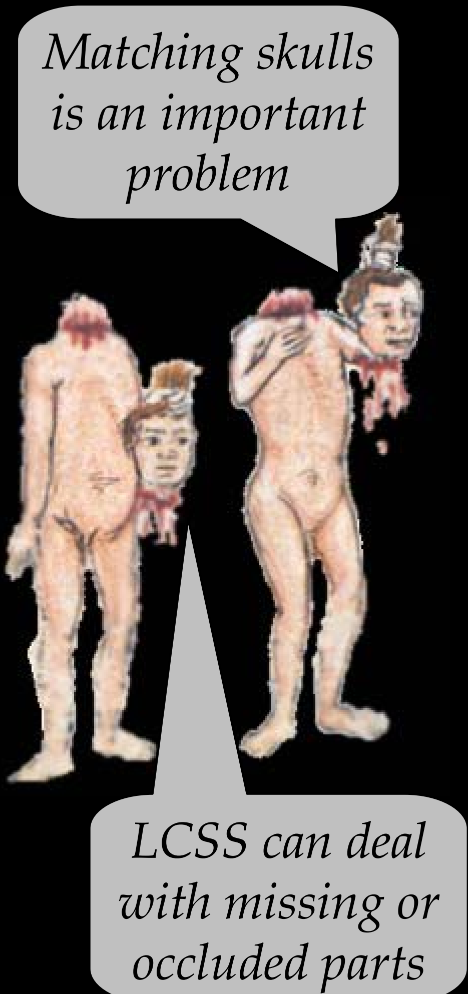
***The Paradiso -- Canto XXIII 28-30**

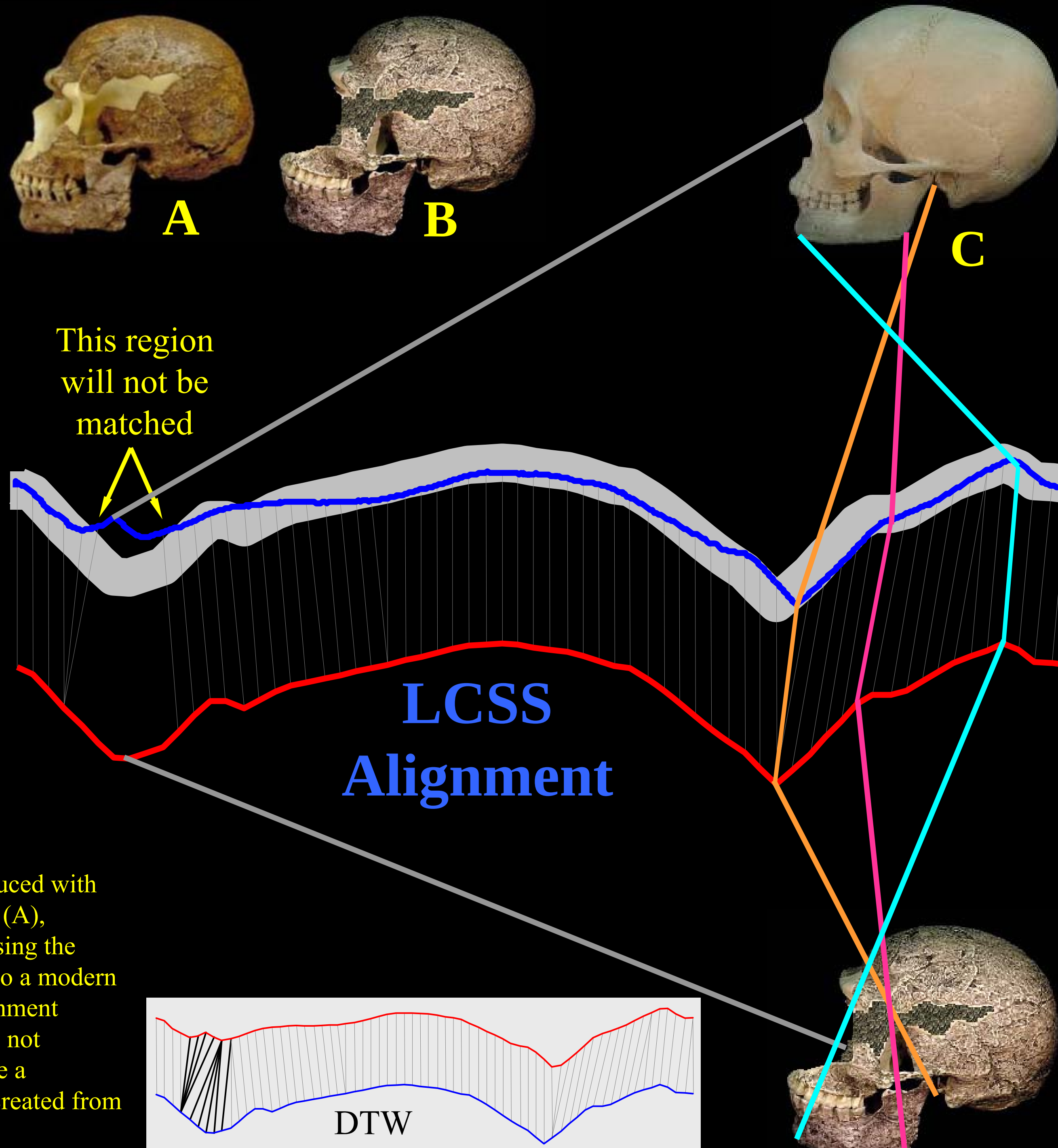
Shape Distance Measures




Euclidean Distance works well for matching many kinds of shapes
Mantled Howler Monkey
Alouatta palliata
Euclidean Distance
Red Howler Monkey
Alouatta seniculus seniculus
LACM
90756

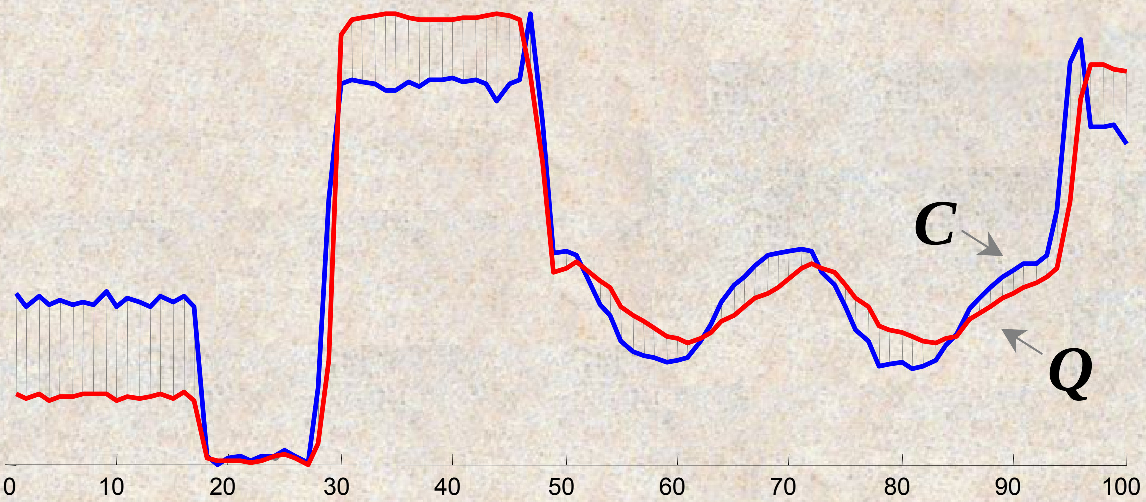

Dynamic Time Warping is useful for natural shapes, which often exhibit intraclass variability
Lowland Gorilla
Gorilla gorilla graueri
DTW
Alignment
Mountain Gorilla
Gorilla gorilla beringei
Is man an ape or an angel?





The famous Skhul V is generally reproduced with the missing bones extrapolated in epoxy (A), however the original Skhul V (**B**) is missing the nose region, which means it will match to a modern human (**C**) poorly, even after DTW alignment (inset). In contrast, LCSS alignment will not attempt to match features that are outside a "matching envelope" (heavy gray line) created from the other sequence.

Euclidean Distance Metric



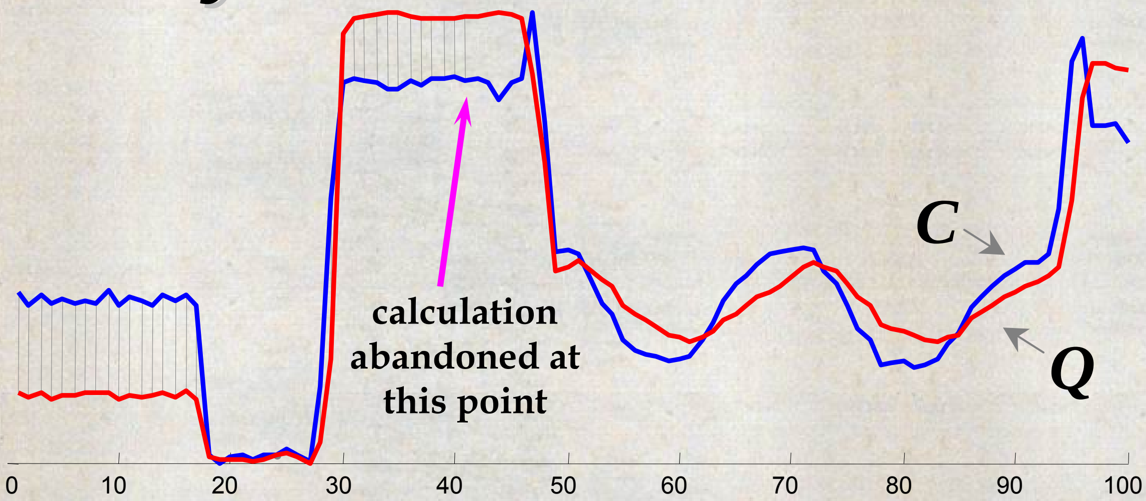
Given two time series $Q = q_1 \ldots q_n$ and $C = c_1 \ldots c_n$, the Euclidean distance between them is defined as:

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

I notice that you Z-normalized the time series first

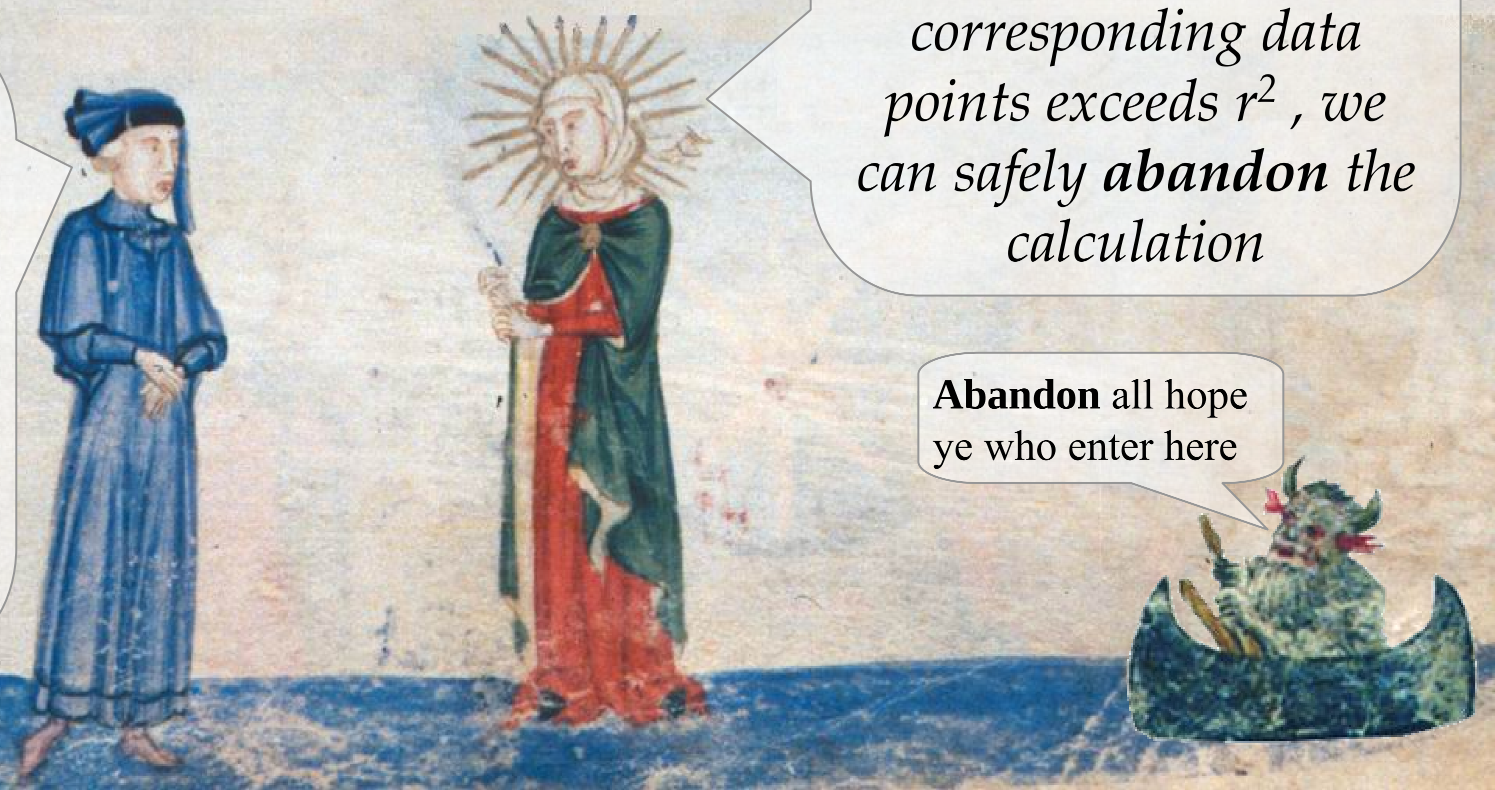
The next slide shows a useful optimization…

Early Abandon Euclidean Distance



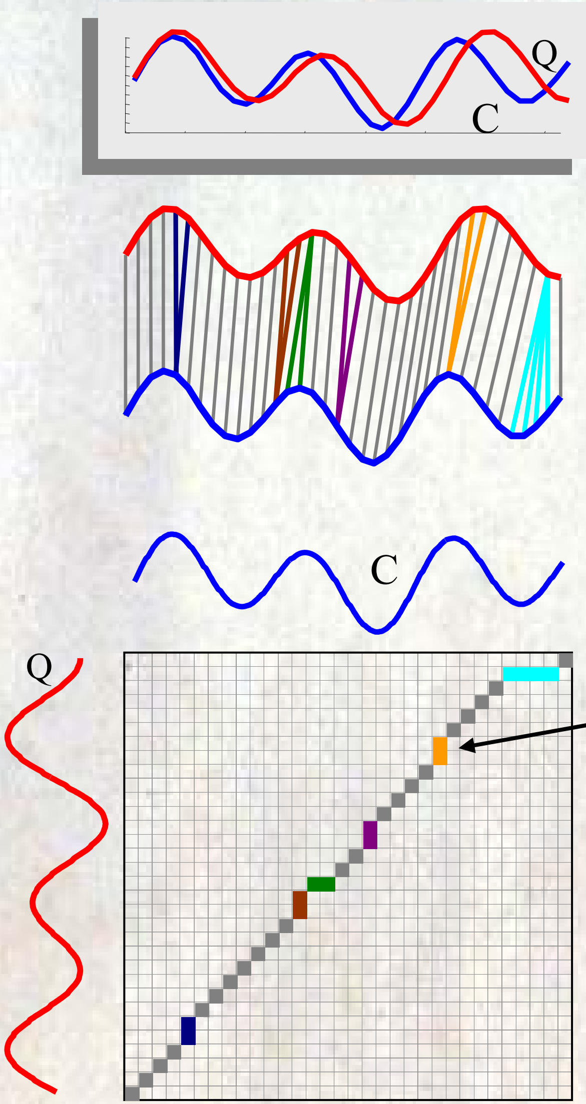
During the computation, if current sum of the squared differences between each pair of corresponding data points exceeds r^2 , we can safely ***abandon*** the calculation

I see, because incremental value is always a lower bound to the final value, once it is greater than the best-so-far, we may as well abandon



Abandon all hope ye who enter here

Dynamic Time Warping I



This is how the DTW alignment is found

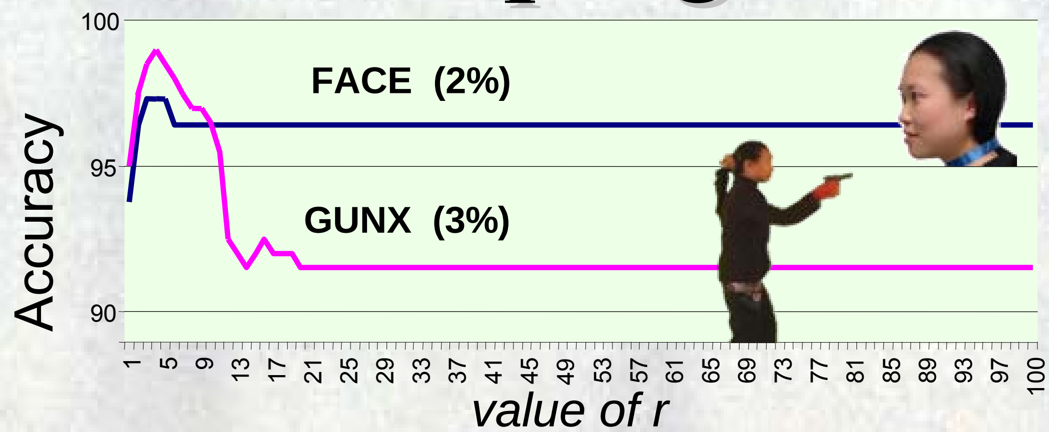
Warping path *w*

$$DTW(Q,C) = \min\left\{ \sqrt{\sum_{k=1}^{K} w_k} \middle/ K \right.$$

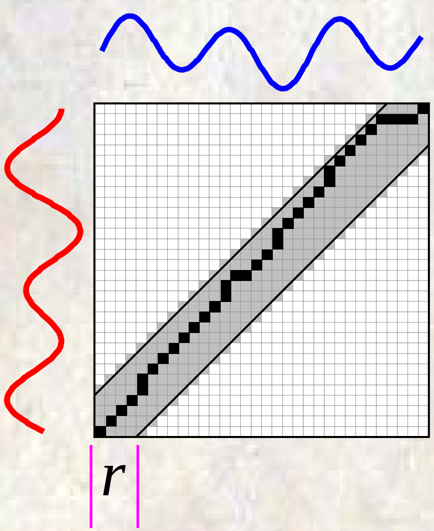
This recursive function gives us the minimum cost path

$$\gamma(i,j) = d(q_i,c_j) + \min\{ \gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1) \}$$

Dynamic Time Warping II



There is an important trick to improve accuracy and speed…



This "constrained warping", together with a lower bounding trick called LB_Keogh can make DTW thousands of times faster! But don't take my word for it...

"LB_Keogh is fast, because it cleverly exploits global constraints…"

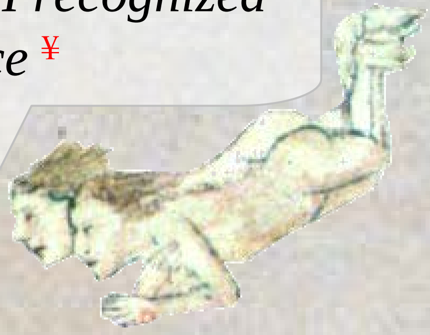
Christos Faloutsos
PODS 2005

See the below for more information about constrained warping:

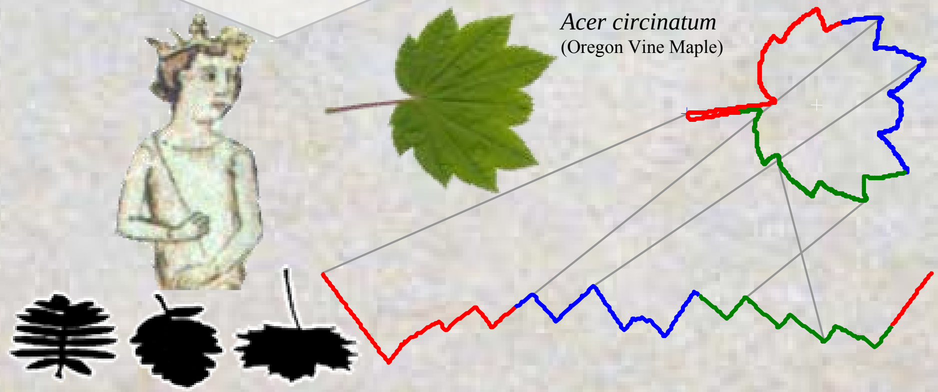
- Xi, Keogh, Shelton, Wei & Ratanamahatana (2006). Fast Time Series Classification Using Numerosity Reduction. ICML
- Ratanamahatana and Keogh. (2004). Everything you know about Dynamic Time Warping is Wrong.

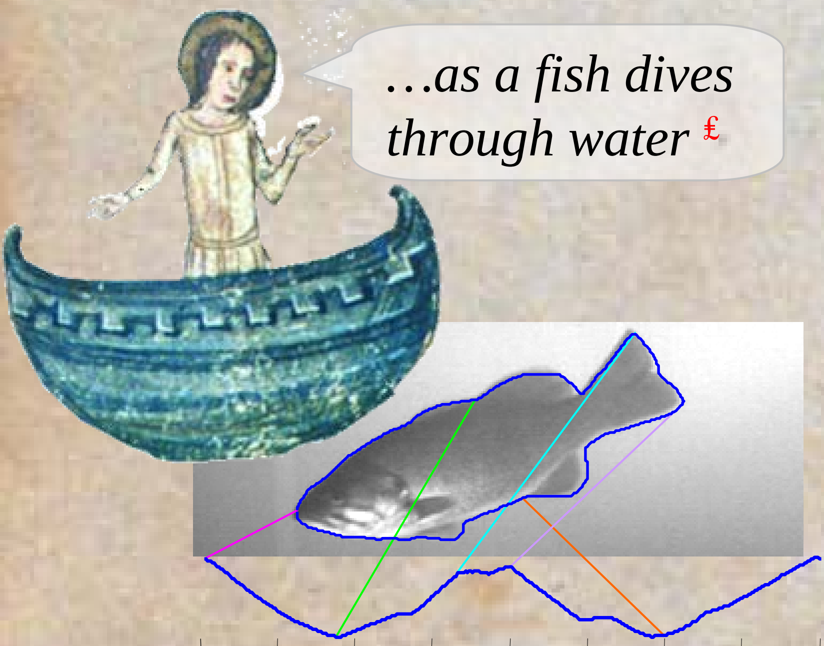
Tests on many diverse datasets

…and I recognized the face ¥



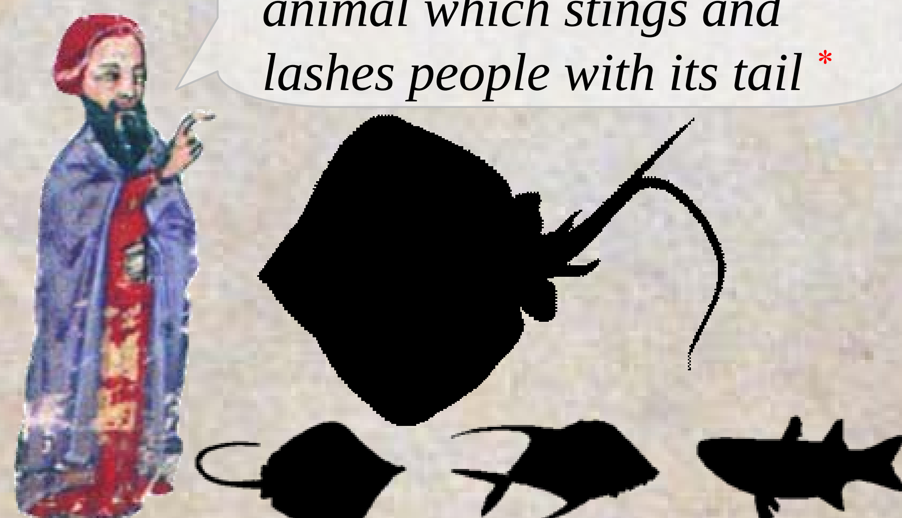
Leaf of mine, in whom I found pleasure ĩ



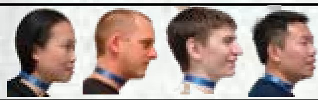
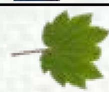
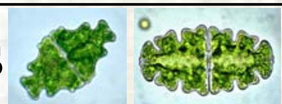
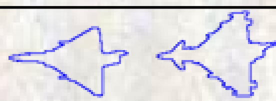


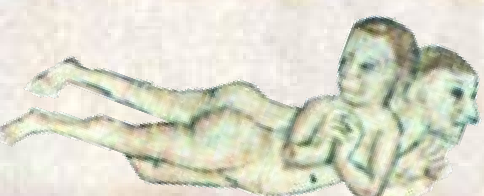
…as a fish dives through water £

…the shape of that cold animal which stings and lashes people with its tail *

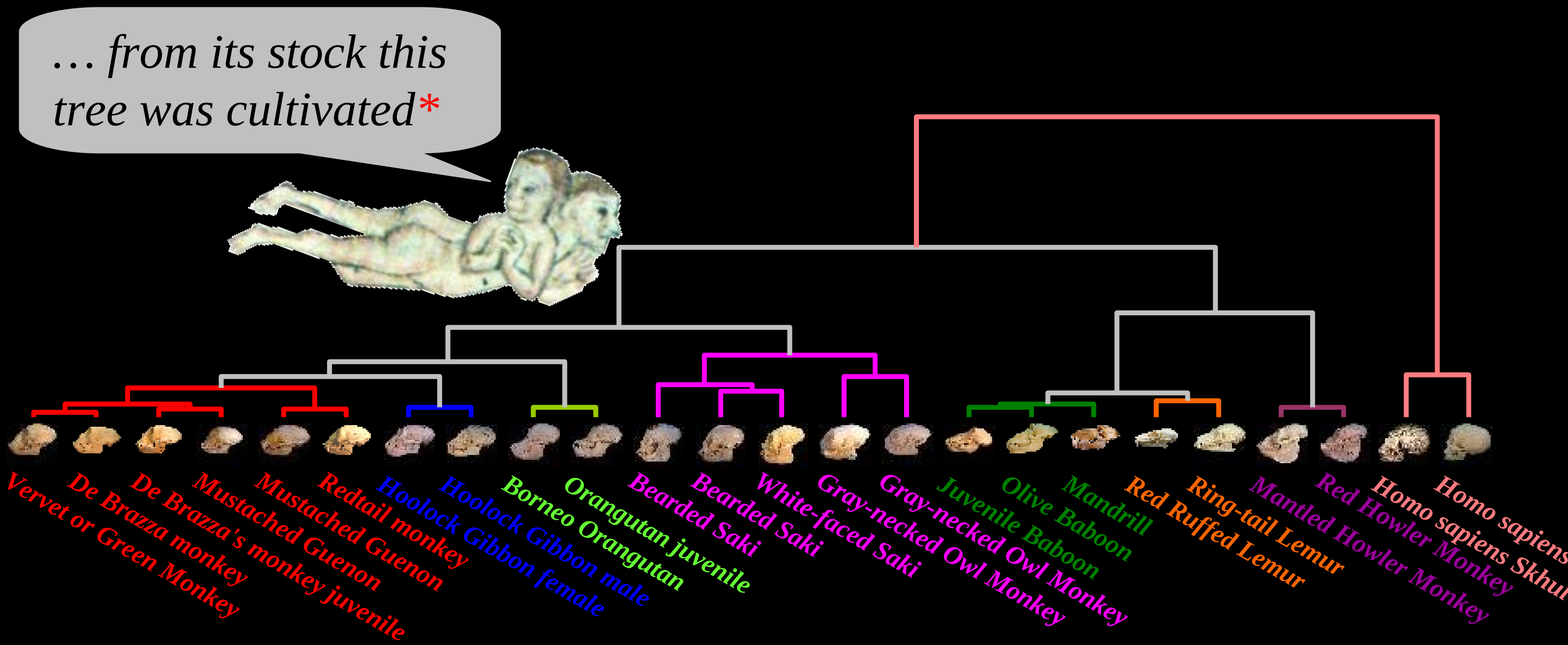


*Purgatorio -- Canto IX 5, ¥Purgatorio -- Canto XXIII, £Purgatorio -- Canto XXVI, ĩParadiso -- Canto XV 88

Name	Classes	Instances	Euclidean Error (%)	DTW Error (%) {r}	Other Techniques
Face 	16	2240	3.839	**3.170**{3}	
Swedish Leaves 	15	1125	13.33	**10.84**{2}	17.82 Söderkvist
Chicken 	5	446	19.96	19.96{1}	20.5 Discrete strings
MixedBag 	9	160	4.375	4.375{1}	Chamfer 6.0, Hausdorff 7.0
OSU Leaves 	6	442	33.71	**15.61**{2}	
Diatoms 	37	781	27.53	27.53{1}	26.0 Morphological Curvature Scale Spaces
Plane 	7	210	0.95	**0.0**{3}	0.55 Markov Descriptor
Fish 	7	350	11.43	**9.71**{1}	36.0 Fourier /Power Cepstrum



Note that DTW is sometimes worth the little extra effort



All these are in the genus *Cercopithecus*, except for the skull identified as being either a Vervet or Green monkey, both of which belong in the Genus of *Chlorocebus* which is in the same Tribe (***Cercopithecini***) as *Cercopithecus*.

Tribe *Cercopithecini*

Cercopithecus

De Brazza's Monkey, *Cercopithecus neglectus*
Mustached Guenon, *Cercopithecus cephus*
Red-tailed Monkey, *Cercopithecus ascanius*

Chlorocebus

Green Monkey, *Chlorocebus sabaceus*
Vervet Monkey, *Chlorocebus pygerythrus*

These are the same species ***Bunopithecus hooloc*** (Hoolock Gibbon)

These are in the Genus ***Pongo***

All these are in the family ***Cebidae***

Family *Cebidae (New World monkeys)*

Subfamily Aotinae

Aotus trivirgatus

Subfamily Pitheciinae sakis

Black Bearded Saki, Chiropotes satanas
White-nosed Saki, Chiropotes albinasus

All these are in the tribe ***Papionini***

Tribe *Papionini*

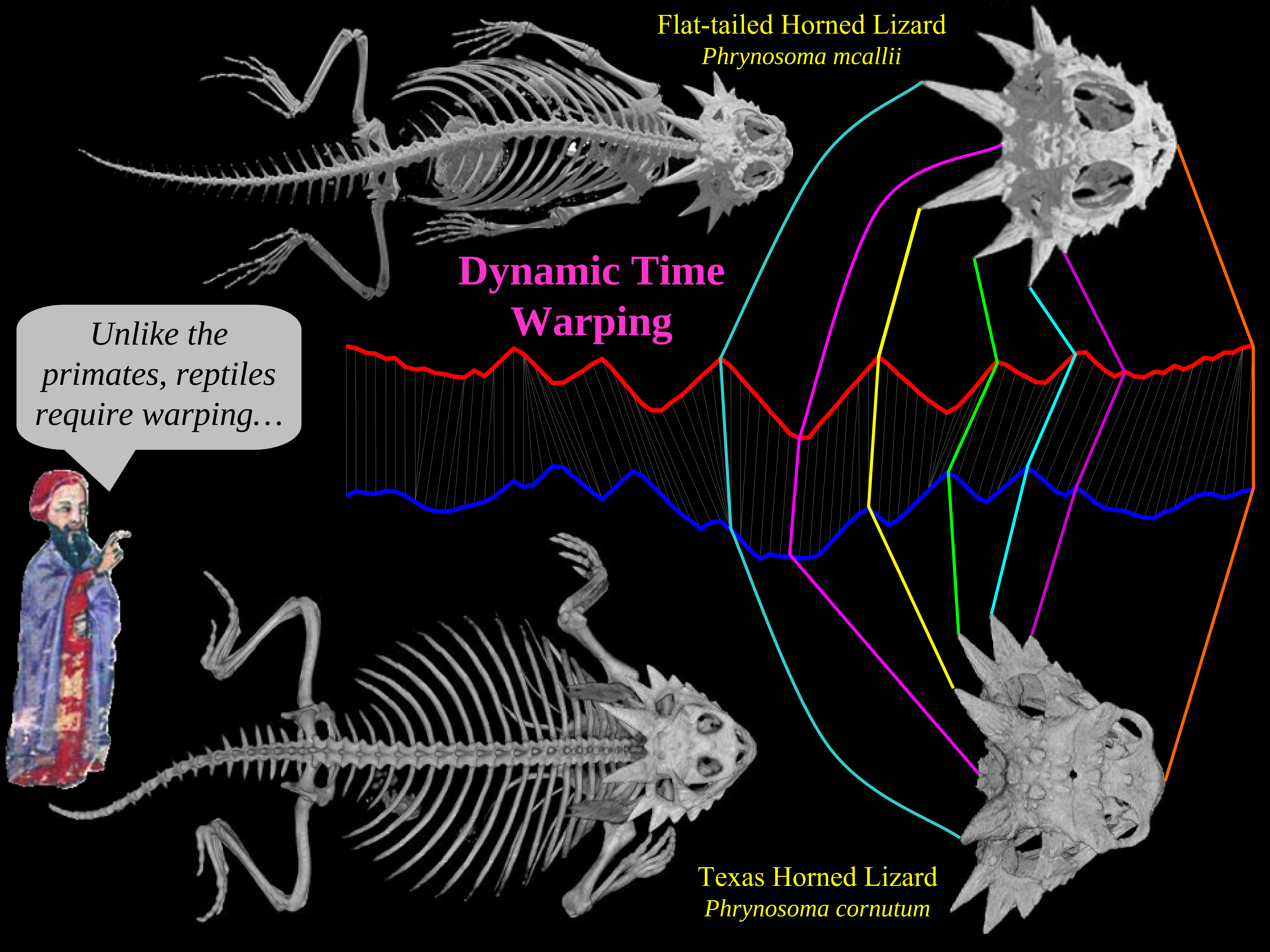
Genus Papio – baboons
Genus Mandrillus- Mandrill

These are in the family ***Lemuridae***

These are in the genus ***Alouatta***

These are in the same species ***Homo sapiens*** (Humans)

***Purgatorio -- Canto XXIV 117**


Flat-tailed Horned Lizard
Phrynosoma mcallii
Dynamic Time Warping
Unlike the primates, reptiles require warping…
Texas Horned Lizard
Phrynosoma cornutum

OK, let us take stock of what we have seen so far

- *There are interesting problems in shape/time series mining (motifs, anomalies, clustering, classification, query-by-content, visualization, joins).*
- *Very simple transformations let us treat shapes as time series.*
- *Very simple distance measures (Euclidean, DTW) work very well.*

We are finally ready to see how symbolic representations, in particular SAX, allow us to solve these problems

Data Mining is Constrained by Disk I/O

For example, suppose you have **one gig** of main memory and want to do K-means clustering…

Clustering ¼ gig of data, 100 sec
Clustering ½ gig of data, 200 sec
Clustering 1 gig of data, 400 sec
Clustering 1.1 gigs of data, 20 hours

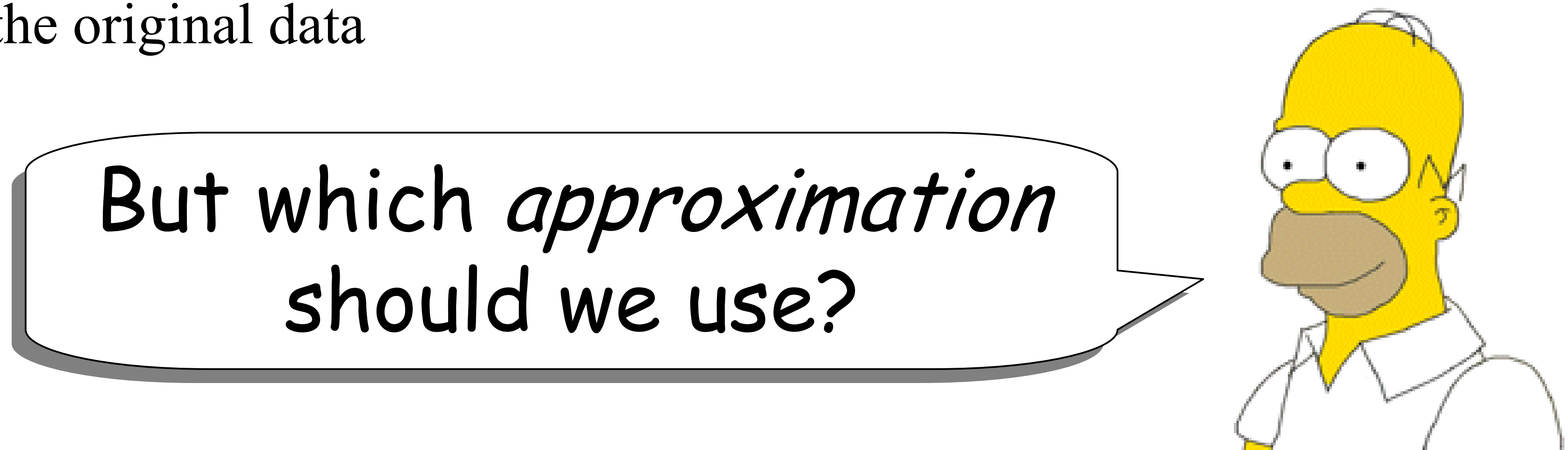
Bradley, M. Fayyad, & Reina: Scaling Clustering Algorithms to Large Databases. KDD 1998: 9-15

The Generic Data Mining Algorithm

- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest

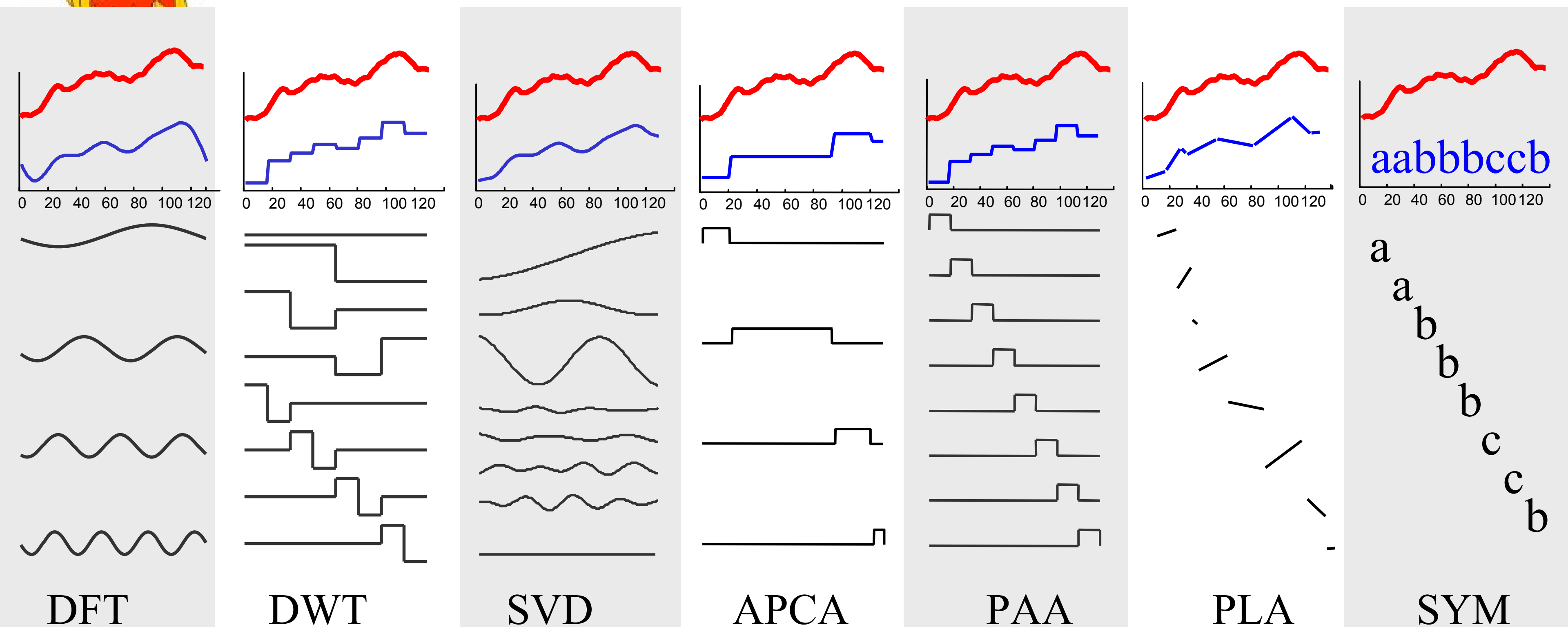
- Approximately solve the problem at hand in main memory

- Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

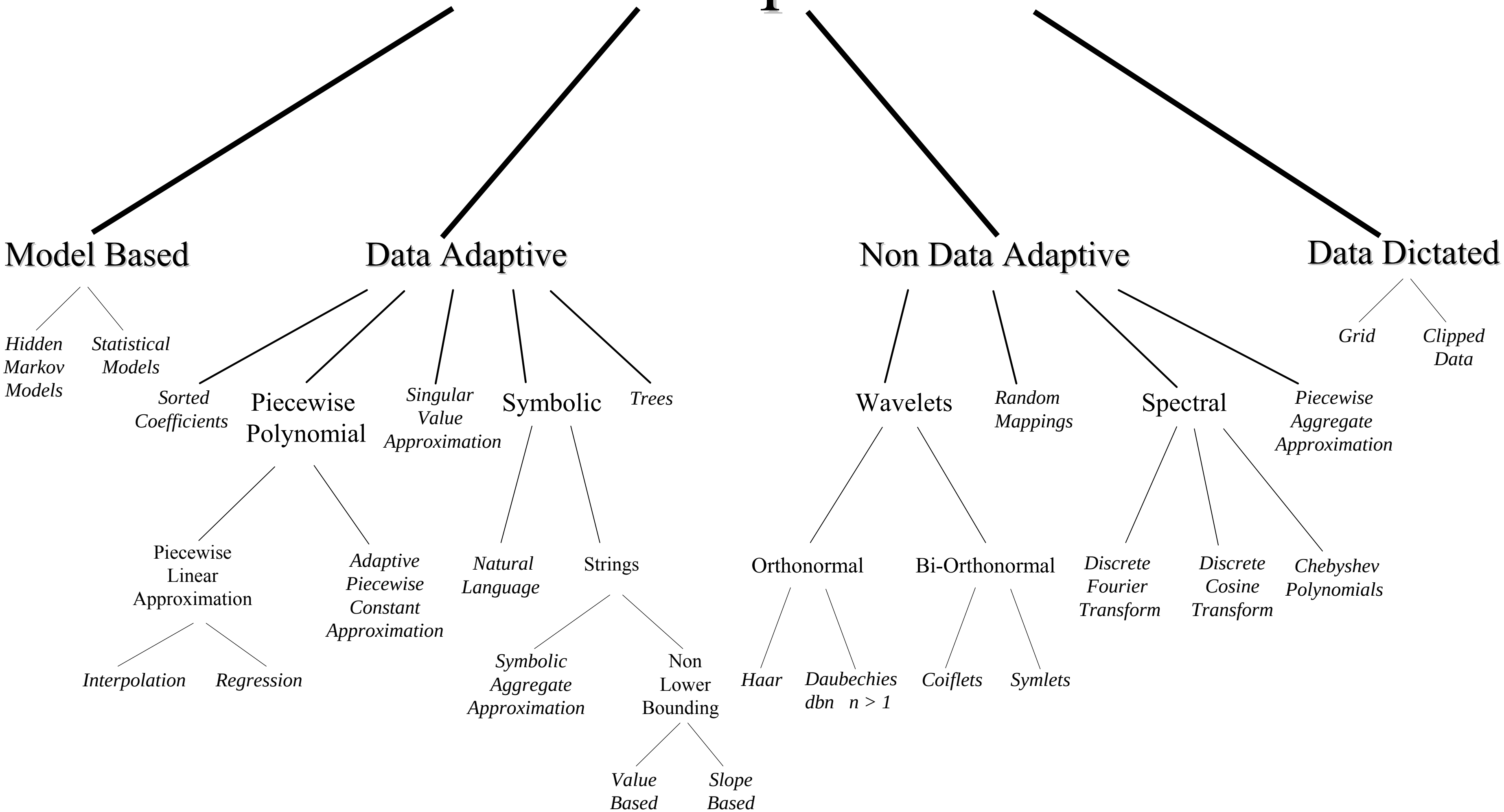


Some approximations of time series...

..note that all except SYM are real valued...



Time Series Representations



The Generic Data Mining Algorithm (revisited)

- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest

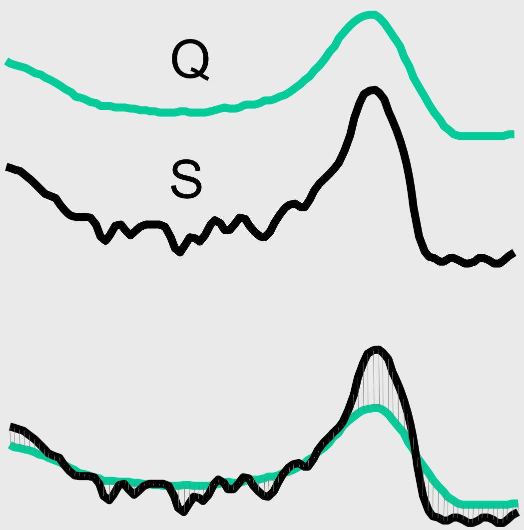
- Approximately solve the problem at hand in main memory

- Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

This *only* works if the approximation allows **lower bounding**

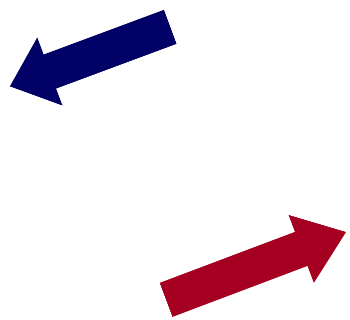
What is Lower Bounding?

• Lower bounding means the estimated distance in the reduced space is always less than or equal to the distance in the original space.

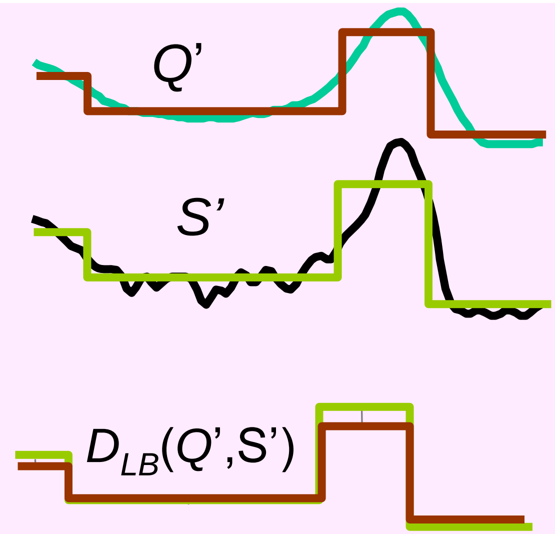


$$D(Q,S) \equiv \sqrt{\sum_{i=1}^{n} (q_i - s_i)^2}$$

Raw Data



Approximation
or
“Representation”



$$D_{LB}(Q',S') \equiv \sqrt{\sum_{i=1}^{M} (sr_i - sr_{i-1})(qv_i - sv_i)^2}$$

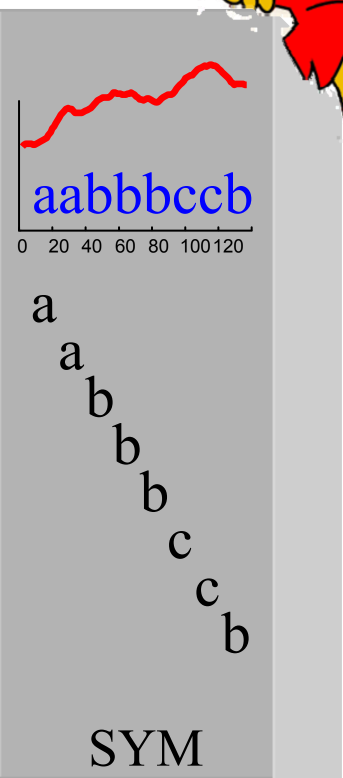
Lower bounding means that for all Q and S, we have: $D_{LB}(Q',S') \leq D(Q,S)$

Lower Bounding functions are known for wavelets, Fourier, SVD, piecewise polynomials, Chebyshev Polynomials and clipped data

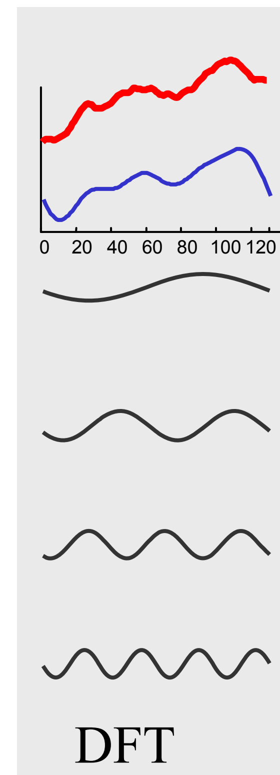
While there are more than 200 different symbolic or discrete ways to approximate time series, none except SAX allows lower bounding

Why do we care so much about *symbolic* representations?

Symbolic Representations Allow:



- Hashing
- Suffix Trees
- Markov Models
- Stealing ideas from text processing/ bioinformatics community
- etc

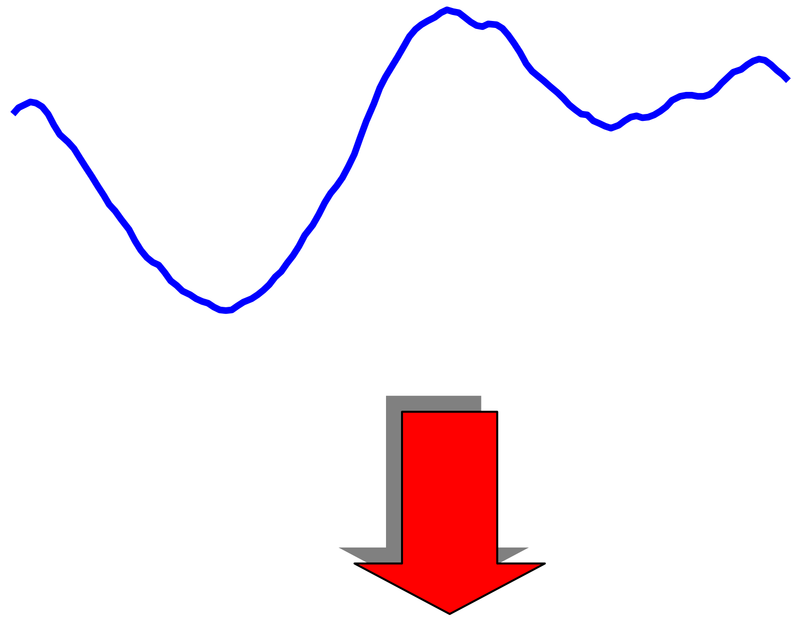


There is *one* symbolic representation of time series, that allows…

- Lower bounding of Euclidean distance
- Lower bounding of the DTW distance
- Dimensionality Reduction
- Numerosity Reduction

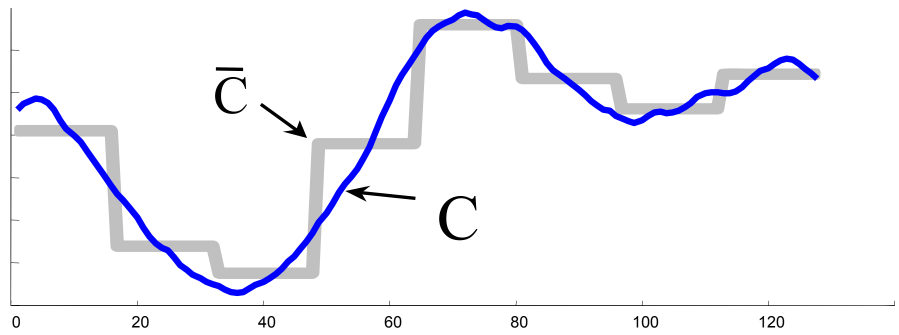
That representation is **SAX**

Symbolic **A**ggregate Appro**X**imation



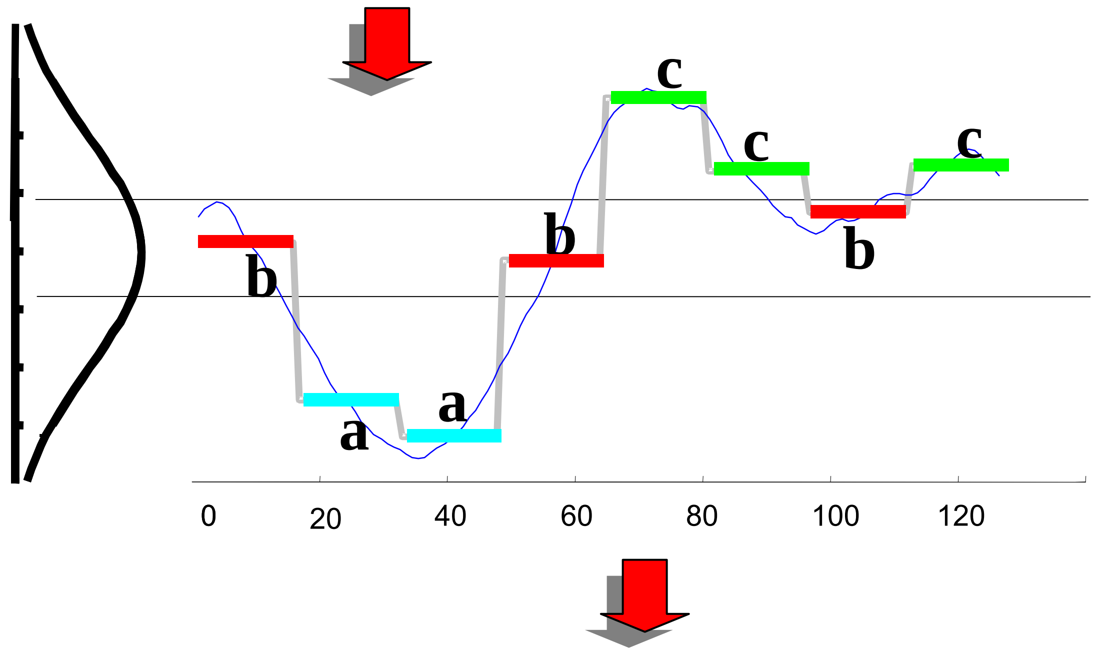
baabccbc

How do we obtain SAX?



First convert the time series to PAA representation, then convert the PAA to symbols

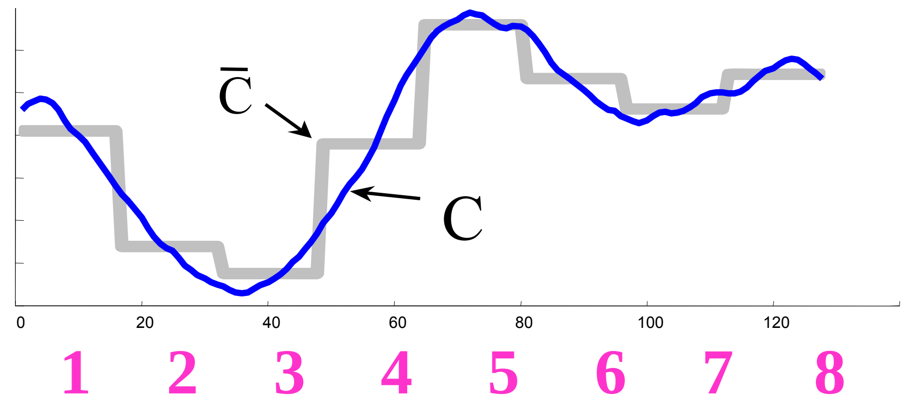
It takes linear time

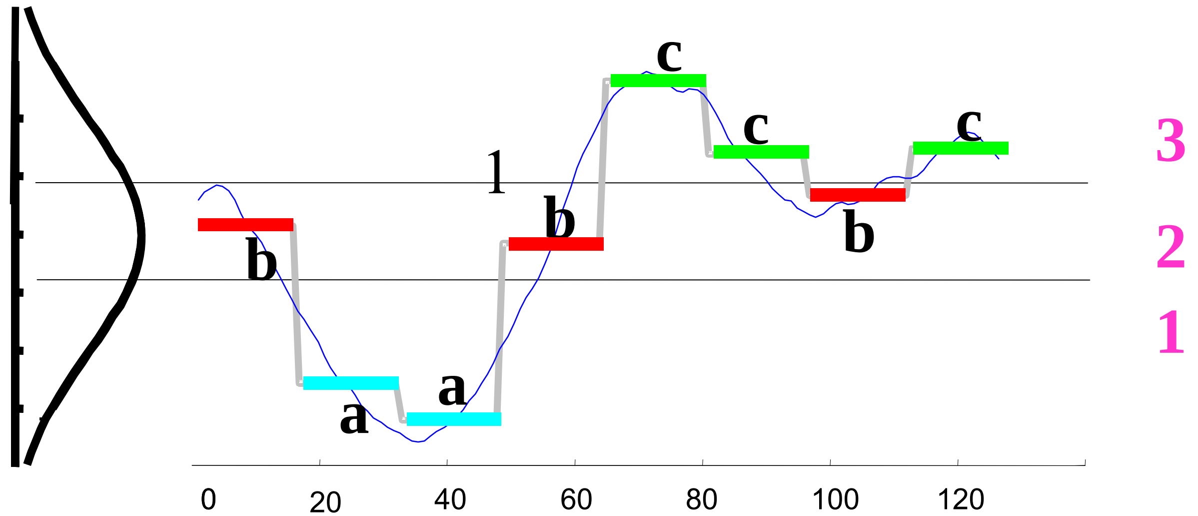


baabccbc

Note we made two parameter choices

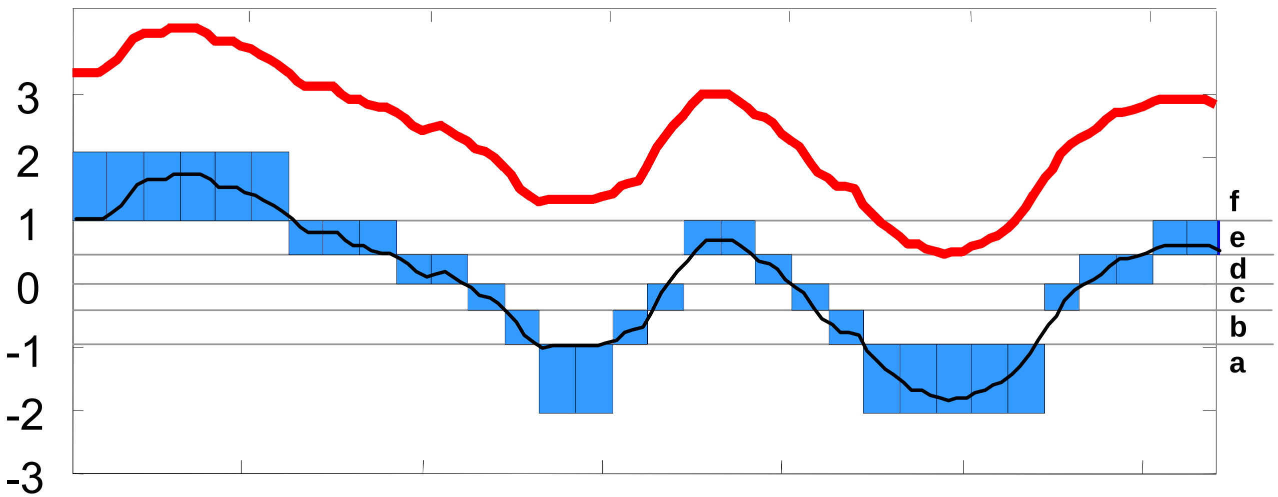
The *word size*, in this case 8.

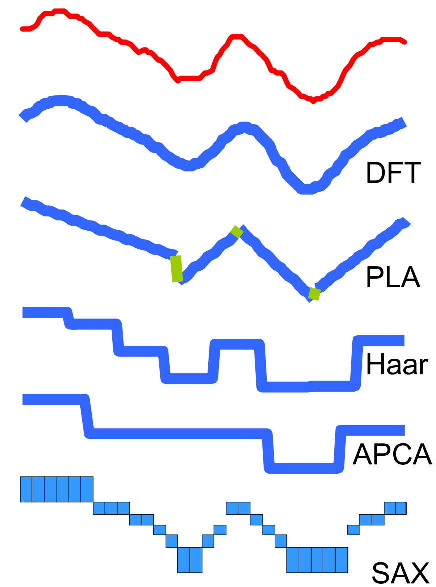




The *alphabet size* (cardinality), in this case 3.

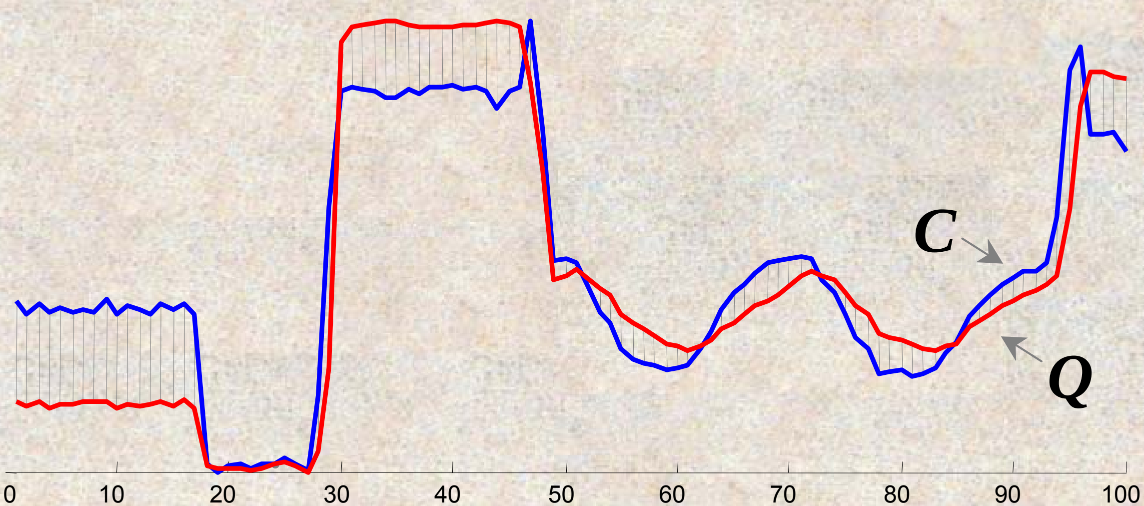
Visual Comparison





- A raw time series of length 128 is transformed into the word "**ffffffeeeddcbaabceedcbaaaacddee**."
  - We can use more symbols to represent the time series since each symbol requires fewer bits than real-numbers (float, double)

SAX Lower Bound to Euclidean Distance Metric



$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

Recall the Euclidean distance?

Yes, here is the function that lower bounds it for SAX, it is called MINDIST

$\hat{C}$ = **bbabcbac**

$\hat{Q}$ = **bbaccbac**

$$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\tfrac{n}{w}} \sqrt{\sum_{i=1}^{w} \left(dist(\hat{q}_i,\hat{c}_i)\right)^2}$$

dist() can be implemented using a table lookup.

dist() table lookup

	a	**b**	**c**
a	0	0	0.67
b	0	0	0
c	0.67	0	0

- *Data mining problems are I/O bound*
- *The generic data mining algorithm mitigates the problem, if you can obey the lower bounding requirement.*
- *There is one approximation of time series that is symbolic and lower bounding, SAX*
- *Being discrete instead of real valued gives SAX some advantages (which we have yet to see)*

OK, let us have another quick review

We are finally ready to see the utility of SAX

Let us consider the utility of SAX for visualizing time series. We start with an apparent digression, visualizing DNA….

TGGCCGTGCTAGGCCCCACCCCTACCTTG
AGTCCCCGCAAGCTCATCTGCGCGAACCA
AACGCCCACCACCCTTGGGTTGAAATTAA
GAGGCGGTTGGCAGCTTCCCAGGCGCACG
ACCTGCGAATAAATAACTGTCCGCACAAG
AGCCCGACGATAGTCGACCCTCTCTAGTCA
CGACCTACACACAGAACCTGTGCTAGACG
CATGAGATAAGCTAACACAAAAACATTTCC
ACTACTGCTGCCCGCGGGCTACCGGCCAC
CCTGGCTCAGCCTGGCGAAGCCGCCCTTC

The DNA of two species…

Are they similar?

CCGTGCTAGGGCCACCTACCTTGGTCC
CCGCAAGCTCATCTGCGCGAACCAGAA
GCCACCACCTTGGGTTGAAATTAAGGA
GCGGTTGGCAGCTTCCAGGCGCACGTA
CTGCGAATAAATAACTGTCCGCACAAG
AGCCGACGATAAAGAAGAGAGTCGACC
CTCTAGTCACGACCTACACACAGAACC
GTGCTAGACGCCATGAGATAAGCTAAC

A	**C**
G	**T**

0.20	0.24
0.26	0.30

CCGTGCTAGGGCCACCTACCTTGGTCC
CCGCAAGCTCATCTGCGCGAACCAGAA
GCCACCACCTTGGGTTGAAATTAAGGA
GCGGTTGGCAGCTTCCAGGCGCACGTA
CTGCGAATAAATAACTGTCCGCACAAG
AGCCGACGATAAAGAAGAGAGTCGACC
CTCTAGTCACGACCTACACACAGAACC
GTGCTAGACGCCATGAGATAAGCTAAC

A	C
G	T

l=1

AA	AC	CA	CC
AG	AT	CG	CT
GA	GC	TA	TC
GG	GT	TG	TT

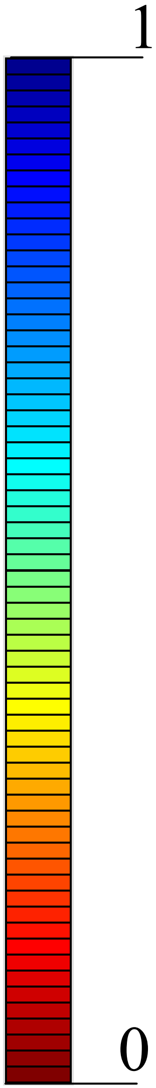
l=2

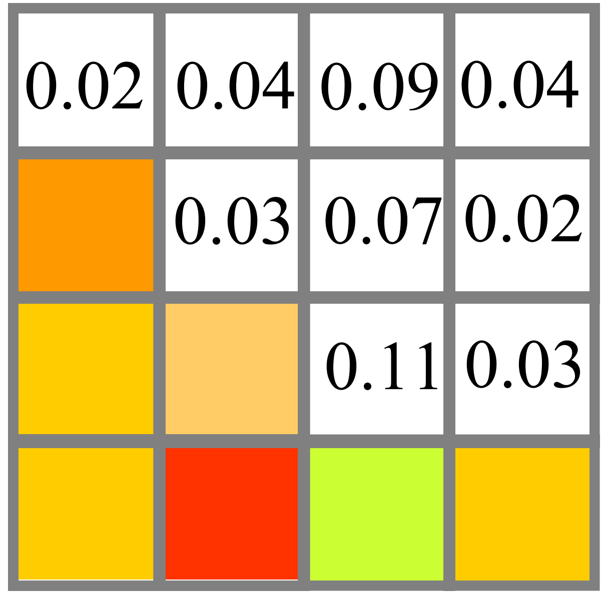
AAA	AAC	ACA	ACC	CAA	CAC	CCA	CCC
AAG	AAT	ACG	ACT	CAG	CAT	CCG	CCT
AGA	AGC	ATA	ATC	CGA	CGC	CTA	CTC
AGG	AGT	ATG	ATT	CGG	CGT	CTG	CTT
GAA	GAC	GCA	GCC	TAA	TAC	TCA	TCC
GAG	GAT	GCG	GCT	TAG	TAT	TCG	TCT
GGA	GGC	GTA	GTC	TGA	TGC	TTA	TTC
GGG	GGT	GTG	GTT	TGG	TGT	TTG	TTT

l=3

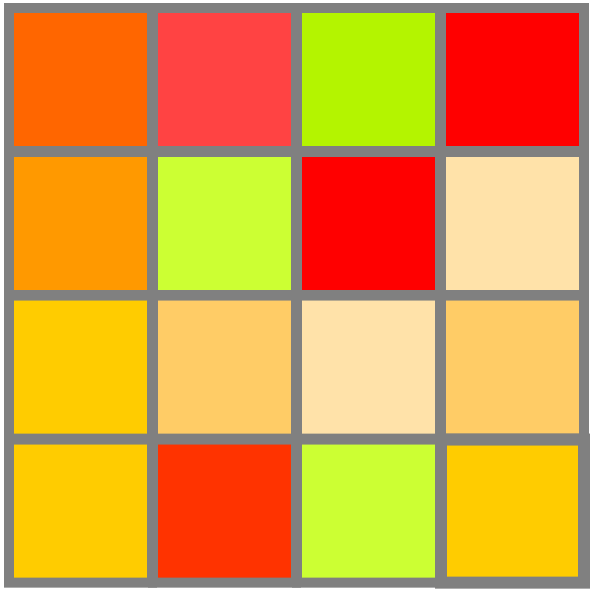
CCGTGCTAGGGCCACCTACCTTGGTCC
CCGCAAGCTCATCTGCGCGAACCAGAA
GCCACCACCTTGGGTTGAAATTAAGGA
GCGGTTGGCAGCTTCCAGGCGCACGTA
CTGCGAATAAATAACTGTCCGCACAAG
AGCCGACGATAAAGAAGAGAGTCGACC
CTCTAGTCACGACCTACACACAGAACC
GTGCTAGACGCCATGAGATAAGCTAAC

l stands for "Level"



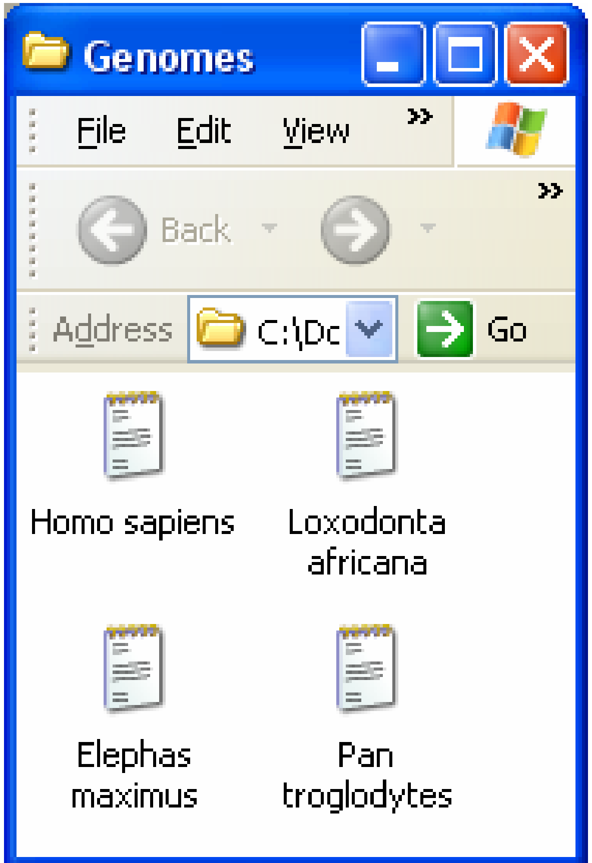


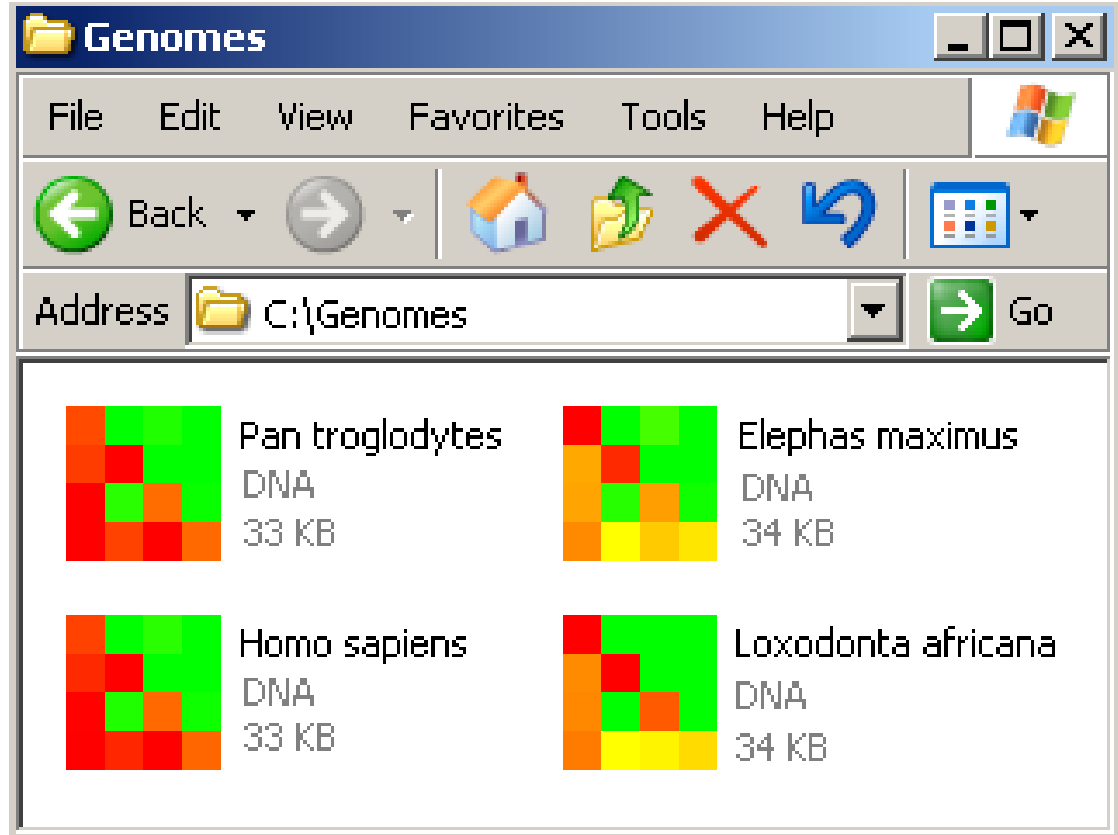
CCGTGCTAGGCCCCACCCCTACCTTGC
GTCCCCGCAAGCTCATCTGCGCGAAC
GAACGCCCACCACCCTTGGGTTGAAA
AAGGAGGCGGTTGGCAGCTTCCCAGG
CACGTACCTGCGAATAAATAACTGTCC
CACAAGGAGCCCGACGATAGTCGACC
CTCTAGTCACGACCTACACACAGAACC
GTGCTAGACGCCATGAGATAAGCTAA

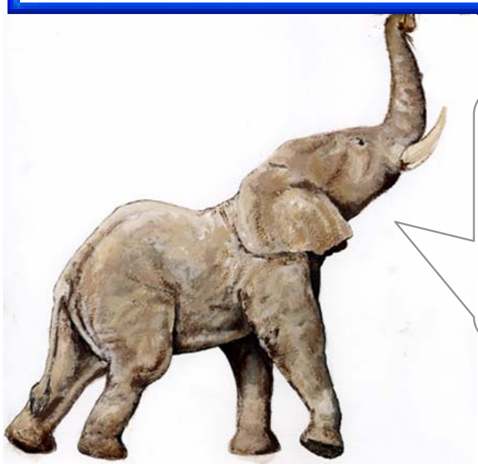


OK. Given any DNA string I can make a colored bitmap, so what?

CCGTGCTAGGCCCCACCCCTACCTTG
GTCCCCGCAAGCTCATCTGCGCGAAC
GAACGCCCACCACCCTTGGGTTGAAA
AAGGAGGCGGTTGGCAGCTTCCCAGG
CACGTACCTGCGAATAAATAACTGTC
CACAAGGAGCCCGACGATAGTCGACC
CTCTAGTCACGACCTACACACAGAAC
GTGCTAGACGCCATGAGATAAGCTAA







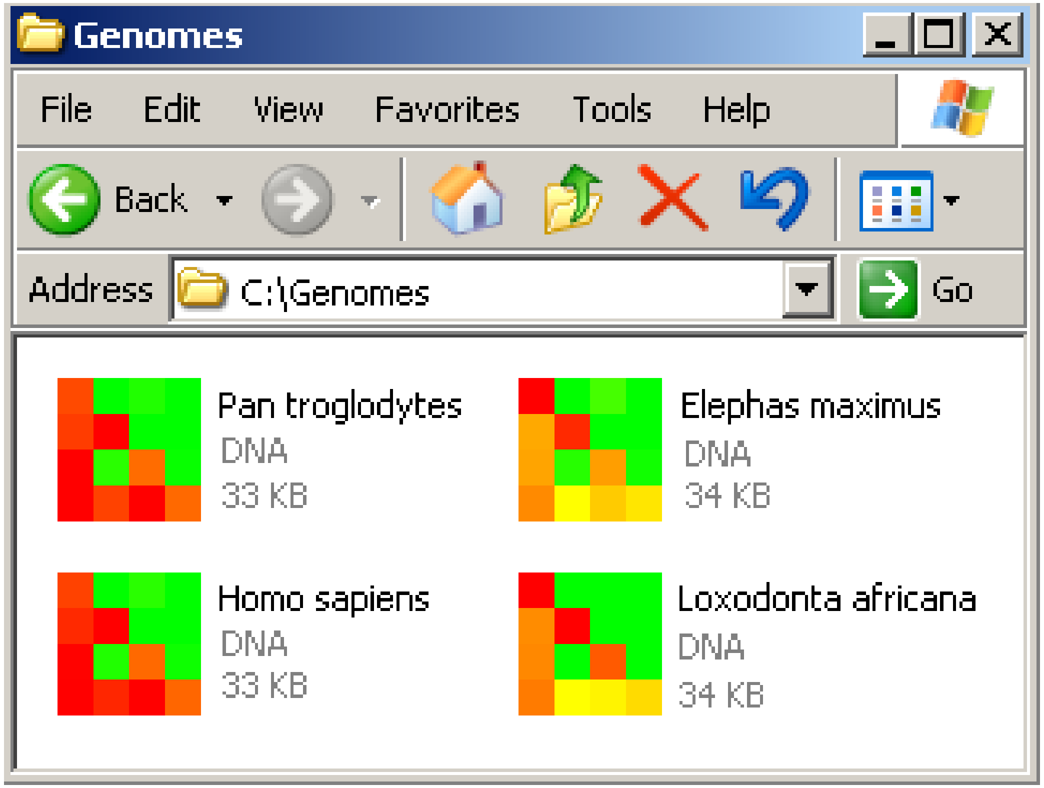
Note *Elephas maximus* is the Indian Elephant, *Loxodonta africana* is the African elephant

Pan troglodytes is the chimpanzee

Two Questions

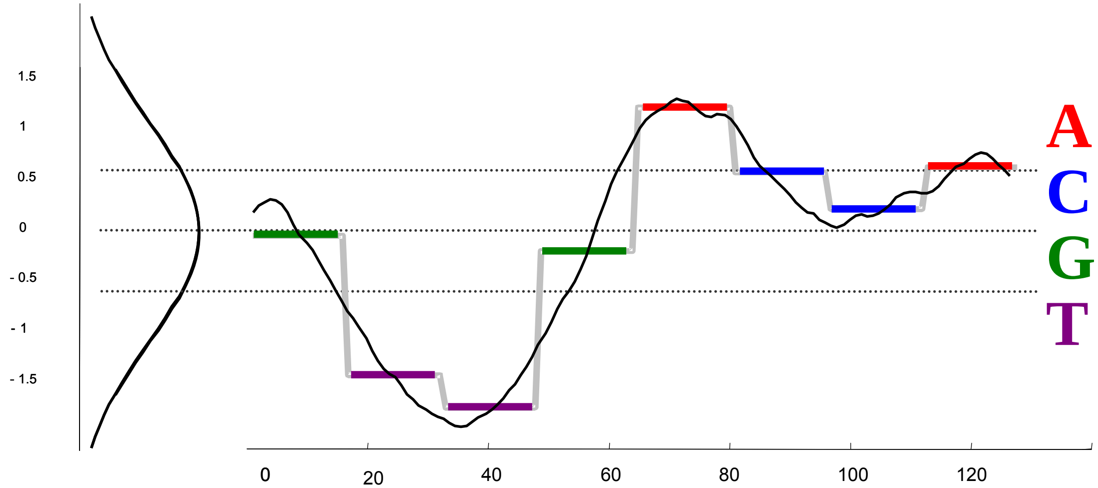
- Can we do something similar for time series?

- Would it be useful?



We call these bitmaps **Intelligent Icons**

Can we make bitmaps for time series?

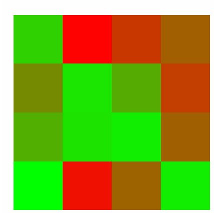


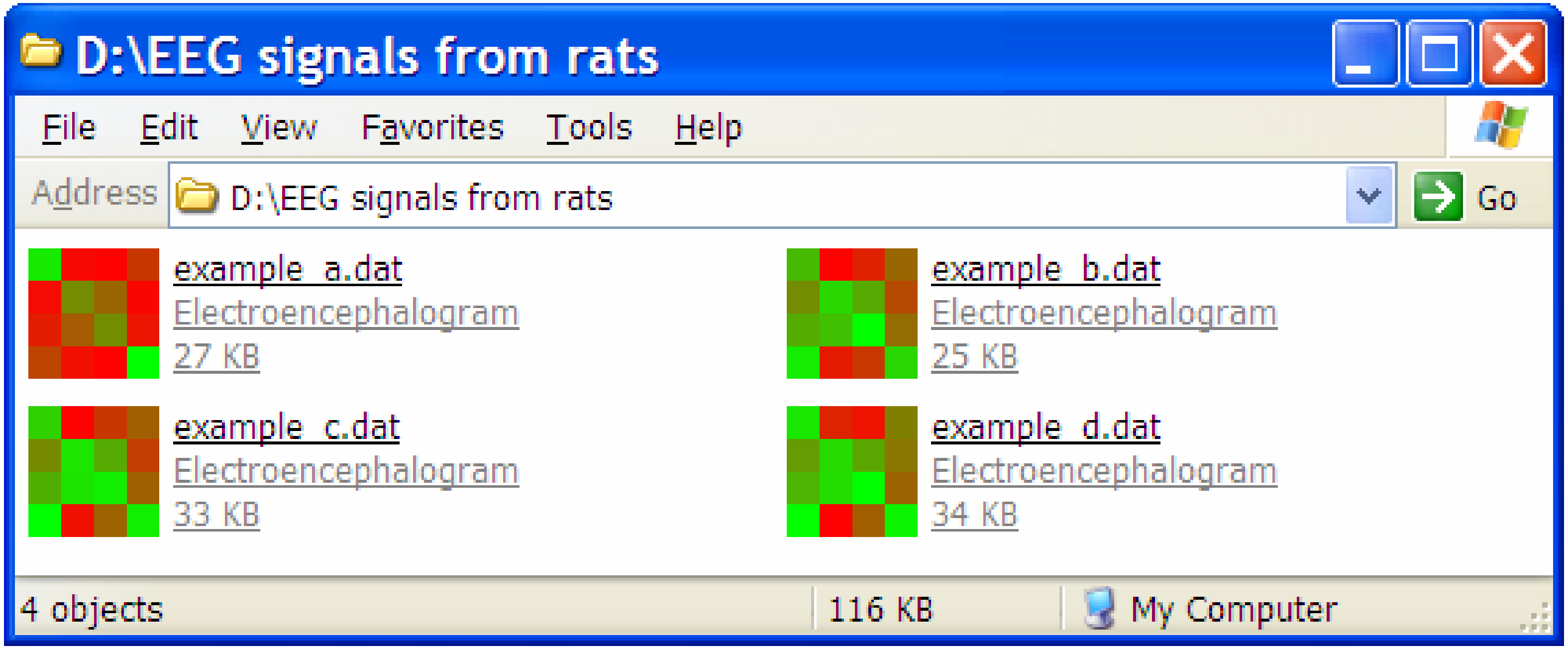
Yes, with SAX!

GTTGACCA

AA	AC	CA	CC
AG	AT	CG	CT
GA	GC	TA	TC
GG	GT	TG	TT

Time Series Bitmap →

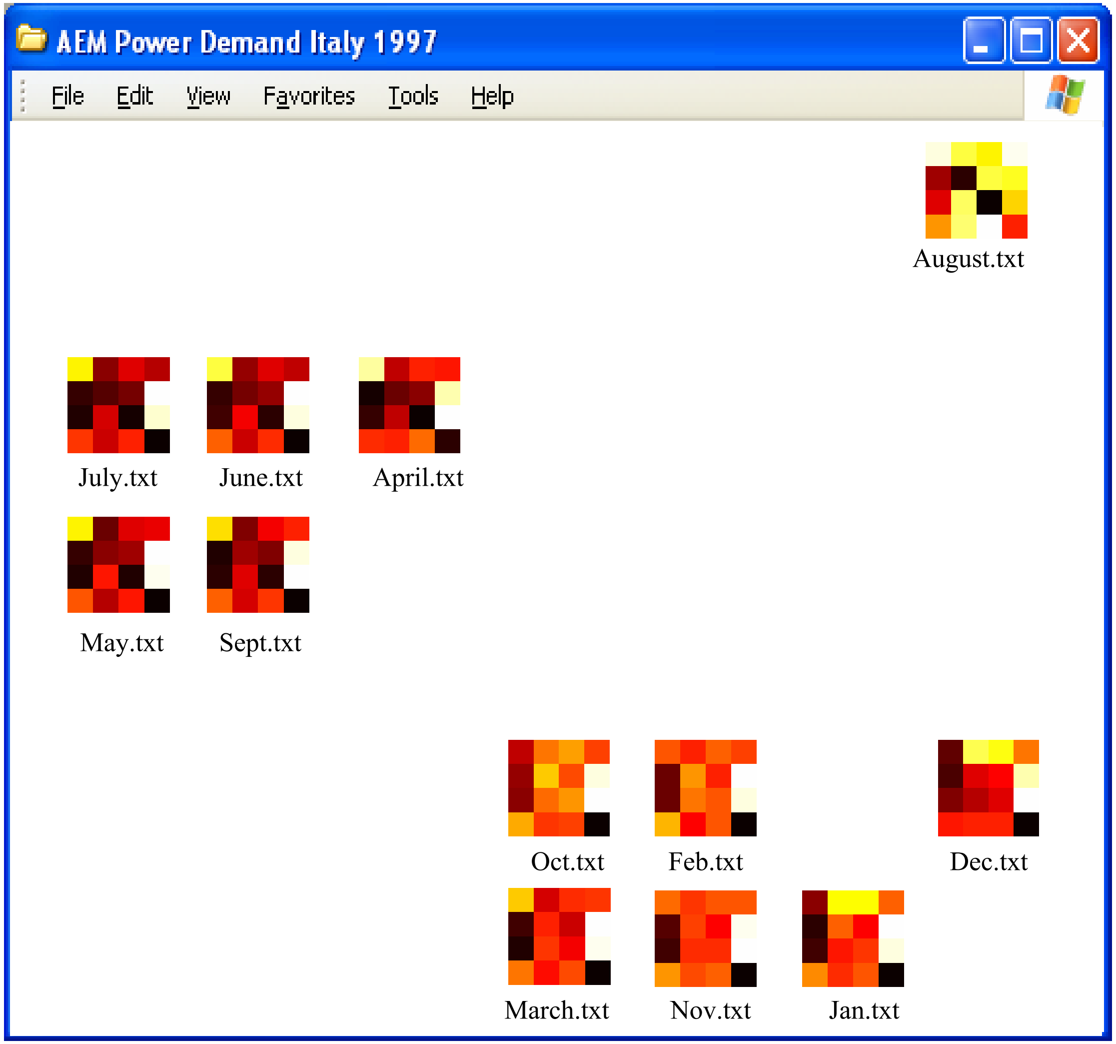


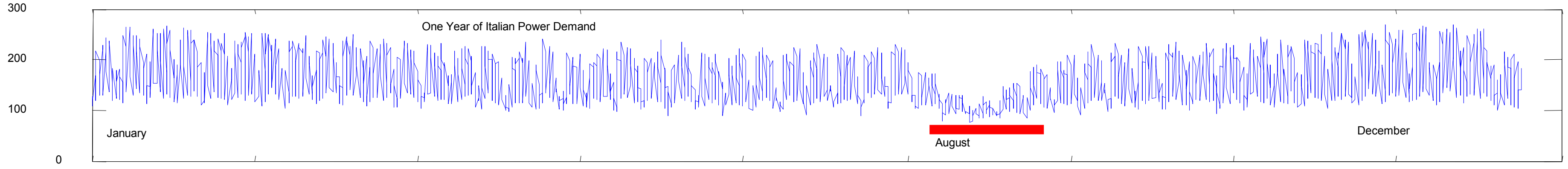


While they are all example of EEGs, *example_a.dat* is from a normal trace, whereas the others contain examples of spike-wave discharges.

We can further enhance the time series bitmaps by arranging the thumbnails by "*cluster*", instead of arranging by *date*, *size*, *name* etc

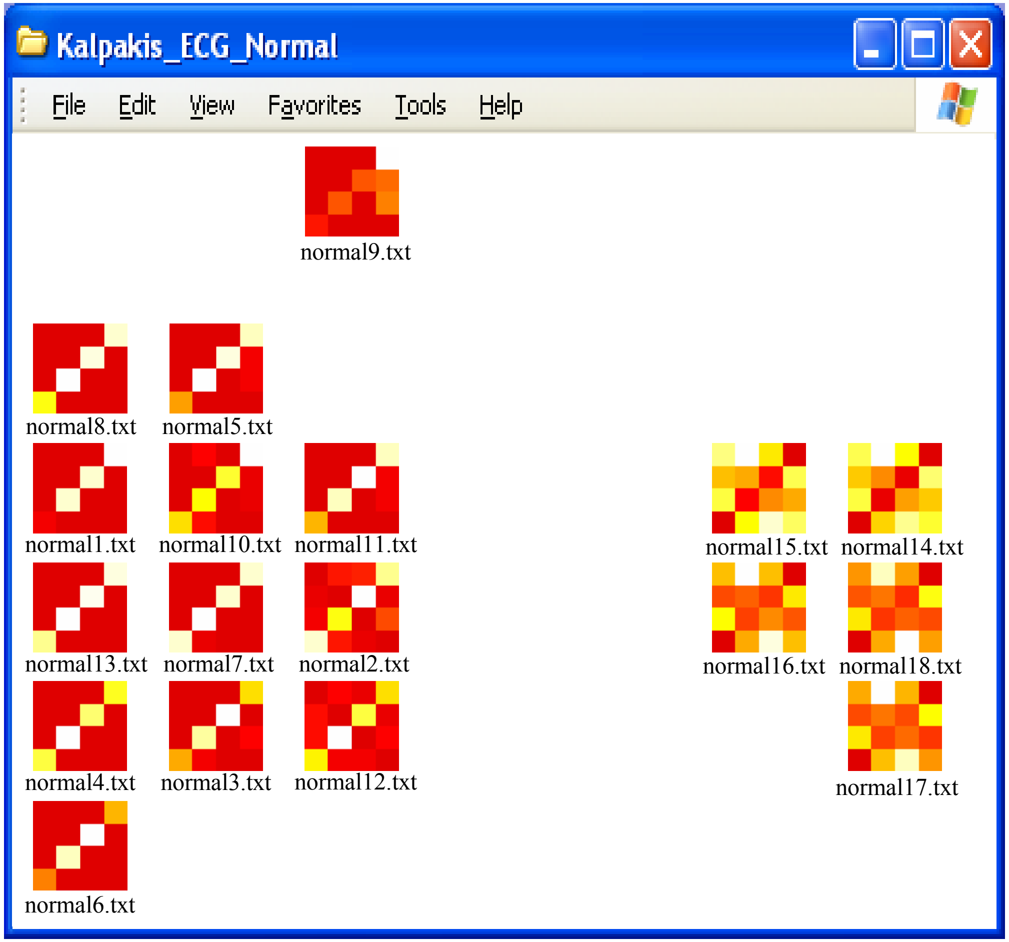
We can achieve this with MDS.

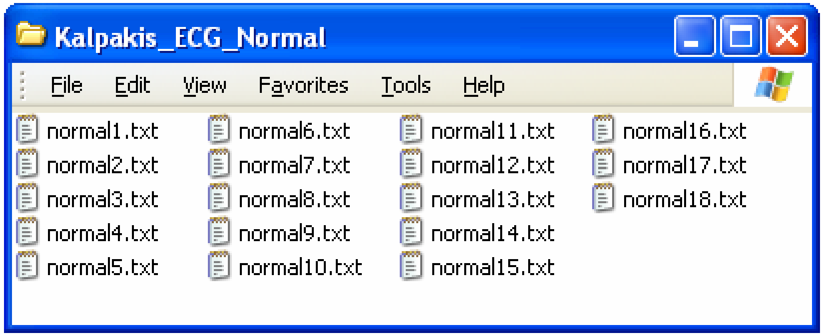


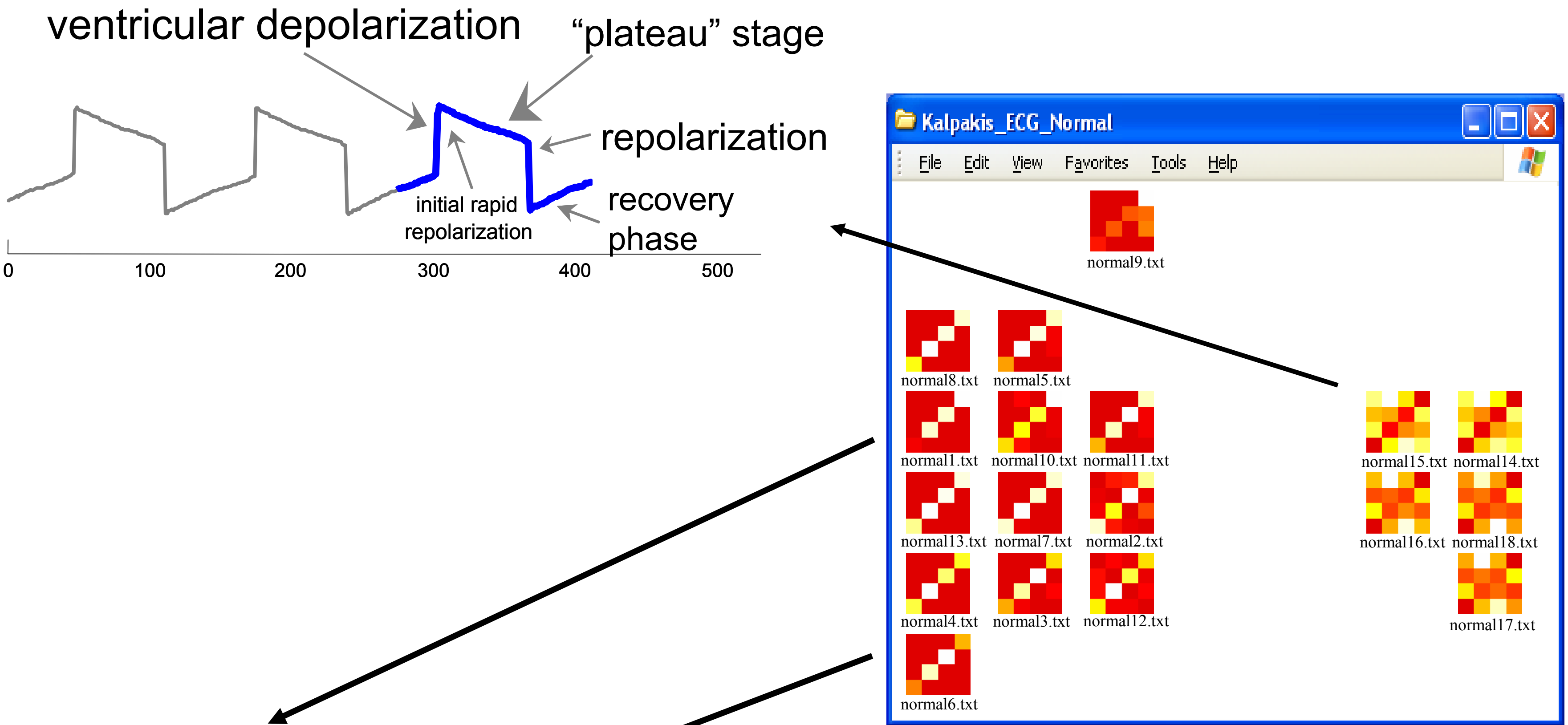


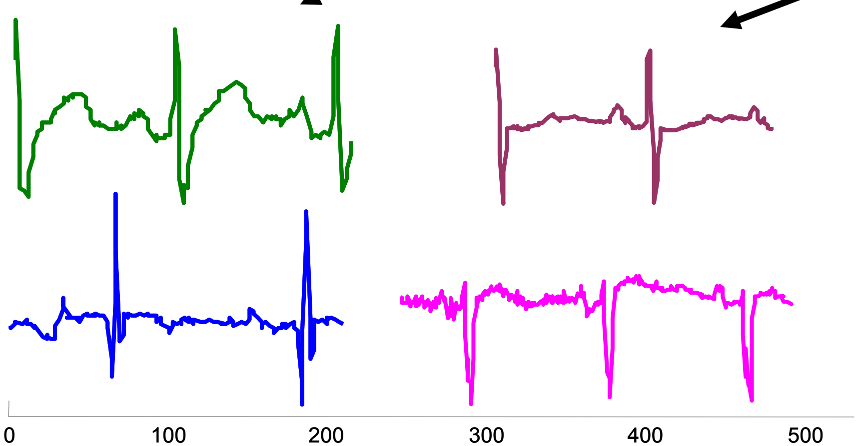
A well known dataset *Kalpakis_ECG*, allegedly contains only ECGS

If we view them as time series bitmaps, a handful stand out…

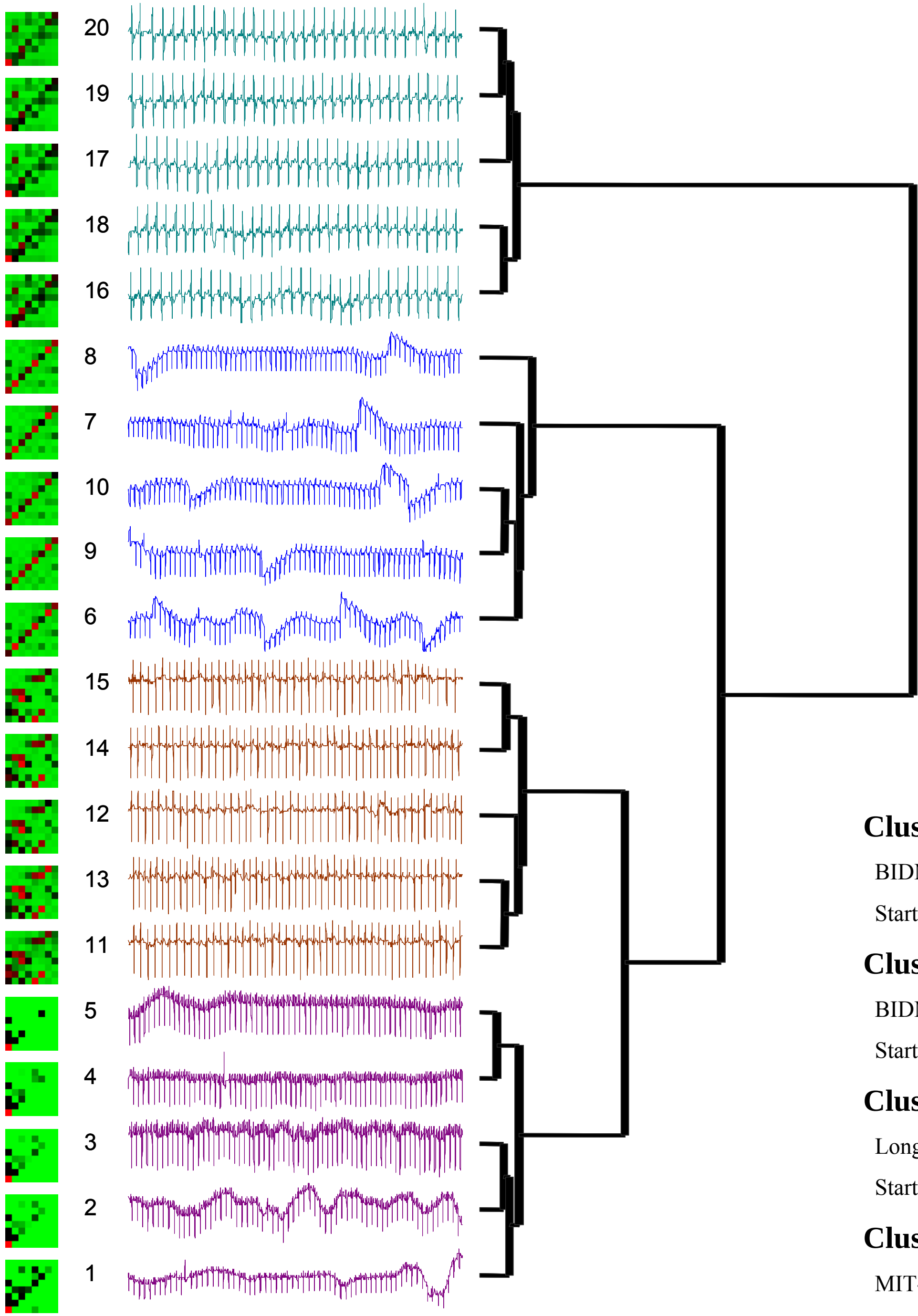








Some of the data are not heartbeats! They are the action potential of a normal pacemaker cell



We can test how much useful information is retained in the bitmaps by using *only* the bitmaps for clustering

Data Key

Cluster 1 (datasets 1 ~ 5):

BIDMC Congestive Heart Failure Database (chfdb): record chf02

Start times at 0, 82, 150, 200, 250, respectively

Cluster 2 (datasets 6 ~ 10):

BIDMC Congestive Heart Failure Database (chfdb): record chf15

Start times at 0, 82, 150, 200, 250, respectively

Cluster 3 (datasets 11 ~ 15):

Long Term ST Database (ltstdb): record 20021

Start times at 0, 50, 100, 150, 200, respectively

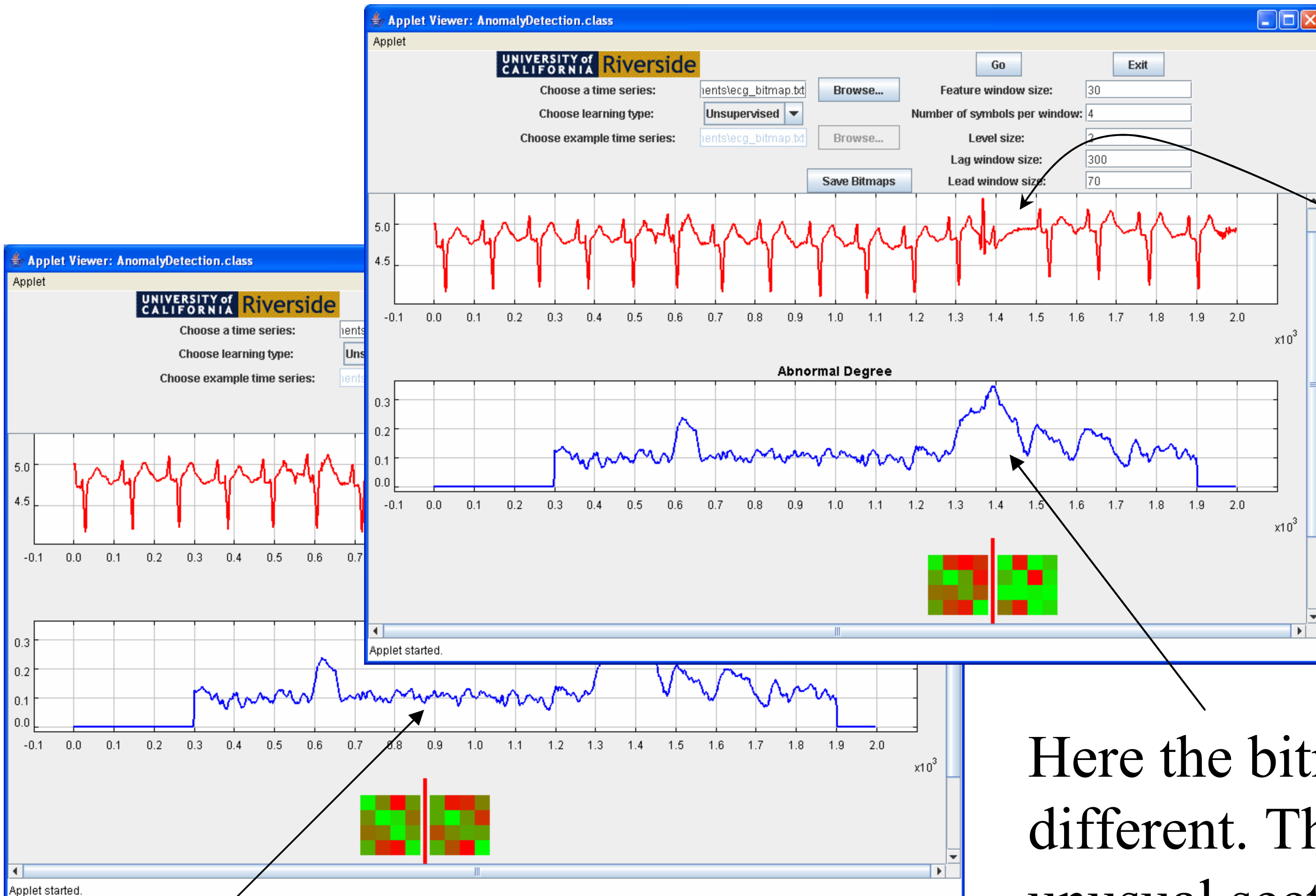
Cluster 4 (datasets 16 ~ 20):

MIT-BIH Noise Stress Test Database (nstdb): record 118e6

Start times at 0, 50, 100, 150, 200, respectively

Lag | *Lead*

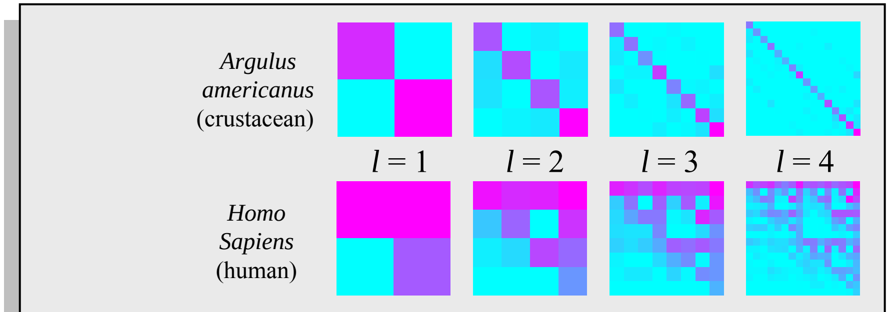
Bitmaps can be used for anomaly detection..



Here is a Premature Ventricular Contraction (PVC)

Here the bitmaps are very different. This is the most unusual section of the time series, and it coincidences with the PVC.

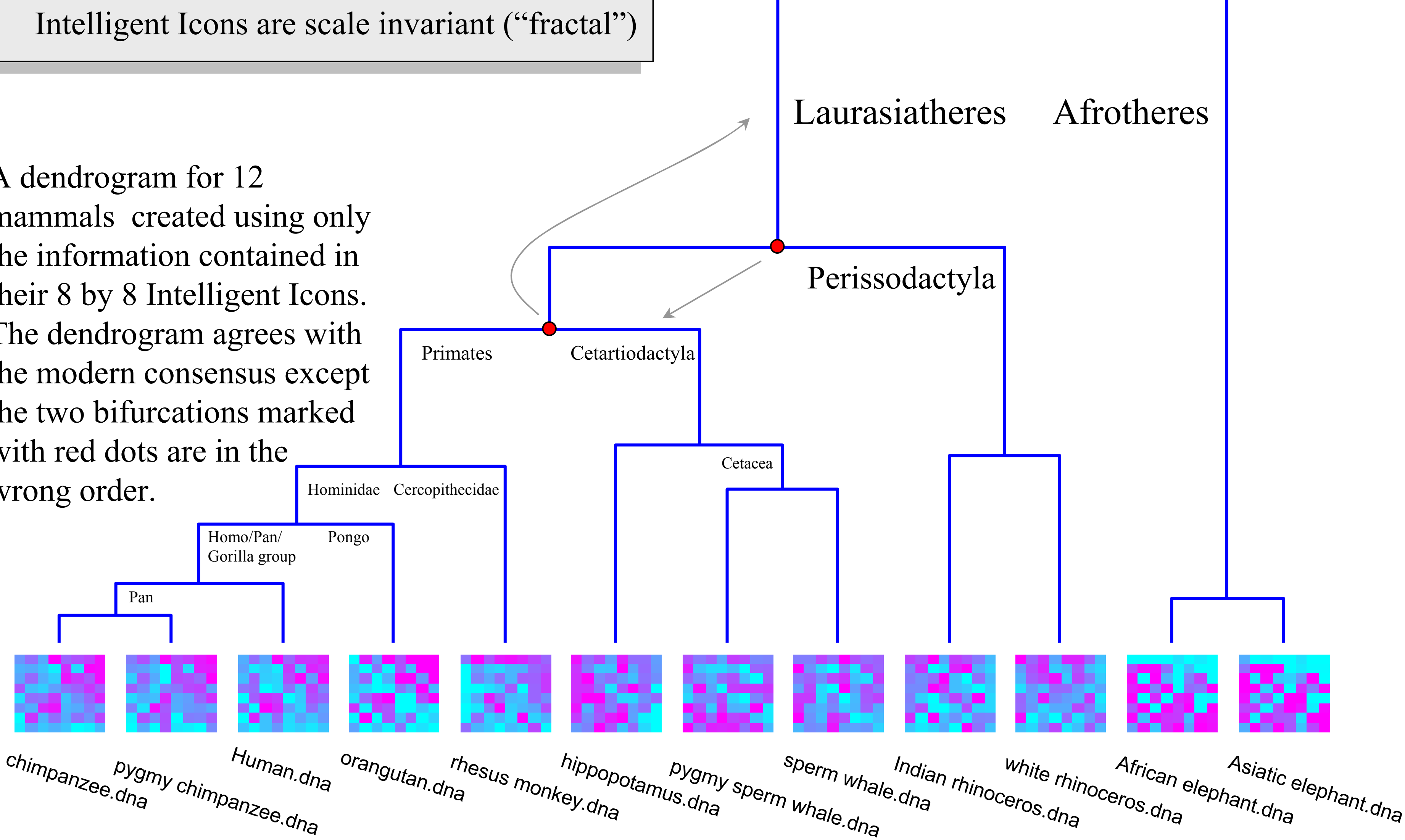
Here the bitmaps are almost the same.



Intelligent Icons are scale invariant (“fractal”)

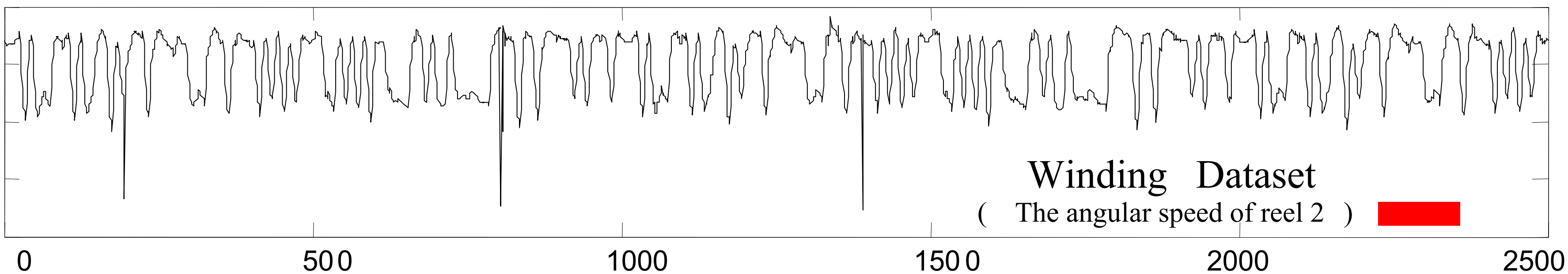
Think of the implications of this, these animals have 3 billion base pairs each, but 64 numbers are enough to cluster them…

A dendrogram for 12 mammals created using only the information contained in their 8 by 8 Intelligent Icons. The dendrogram agrees with the modern consensus except the two bifurcations marked with red dots are in the wrong order.



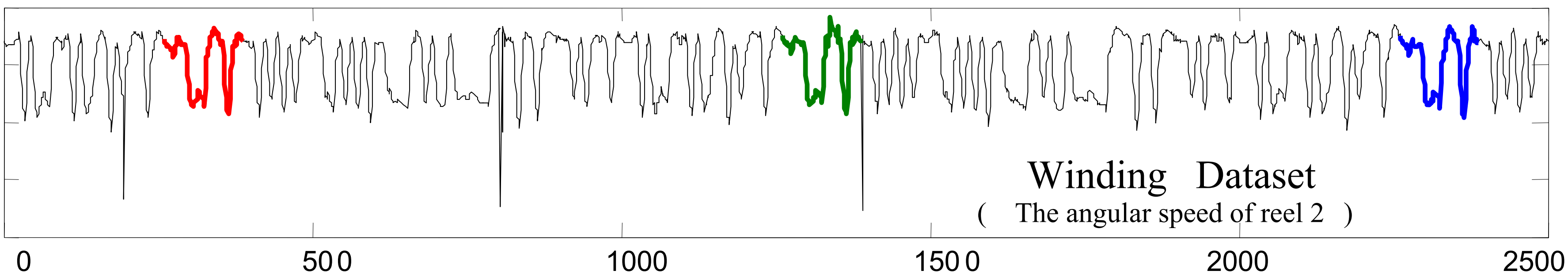
Time Series Motif Discovery

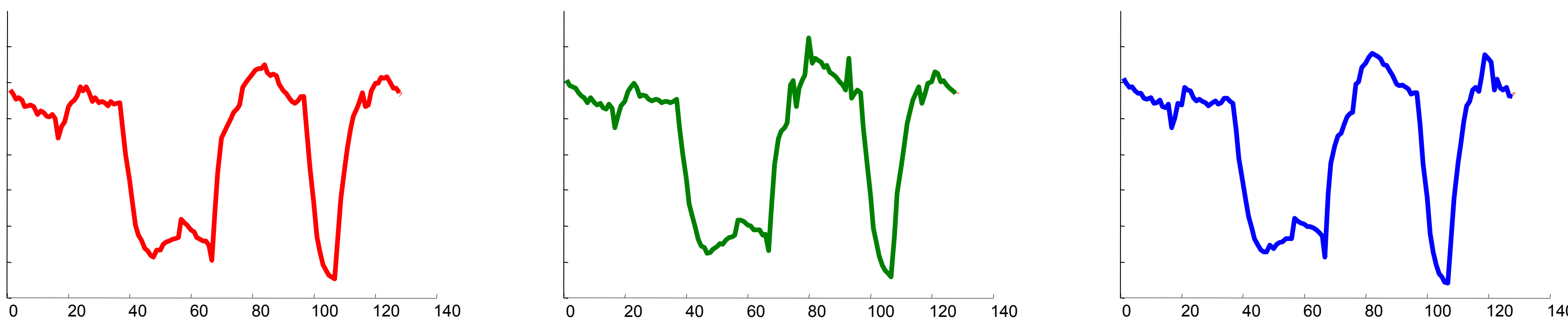
(finding repeated patterns)



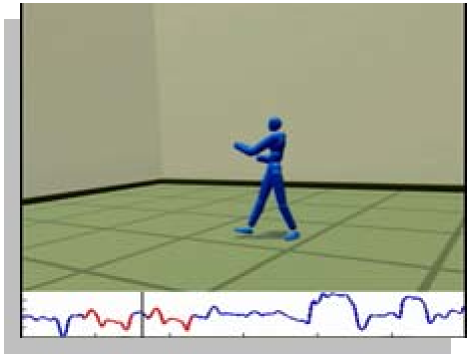
Are there any repeated patterns, of about this length in the above time series?

Time Series Motif Discovery
(finding repeated patterns)





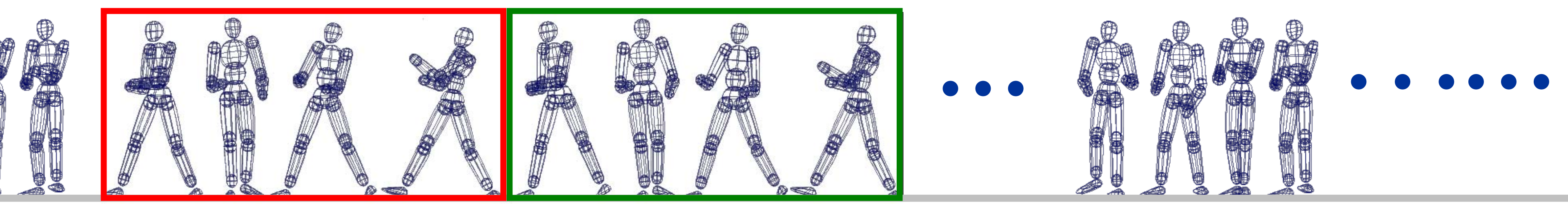
Why Find Motifs? I

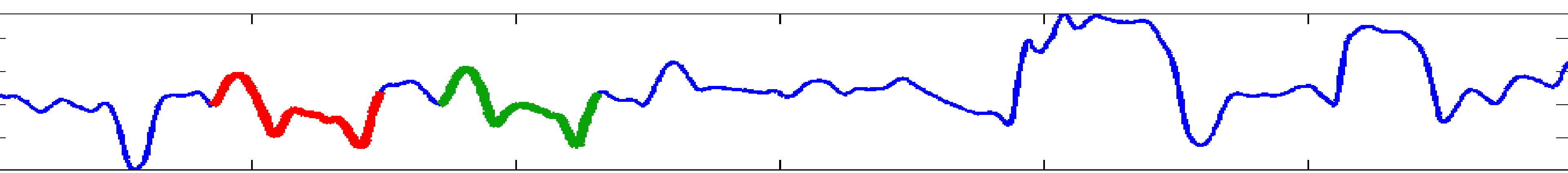


To see the full video go to..
www.cs.ucr.edu/~eamonn/SIGKDD07/UniformScaling.html
Or search YouTube for "Time series motifs "

Finding motifs in motion capture allows efficient editing of special effects, and can be used to allow more natural interactions with video games…

- Tanaka, Y. & Uehara, K.
- Araki , Arita and Taniguchi
- Celly, B. & Zordan, V. B.





Why Find Motifs? II

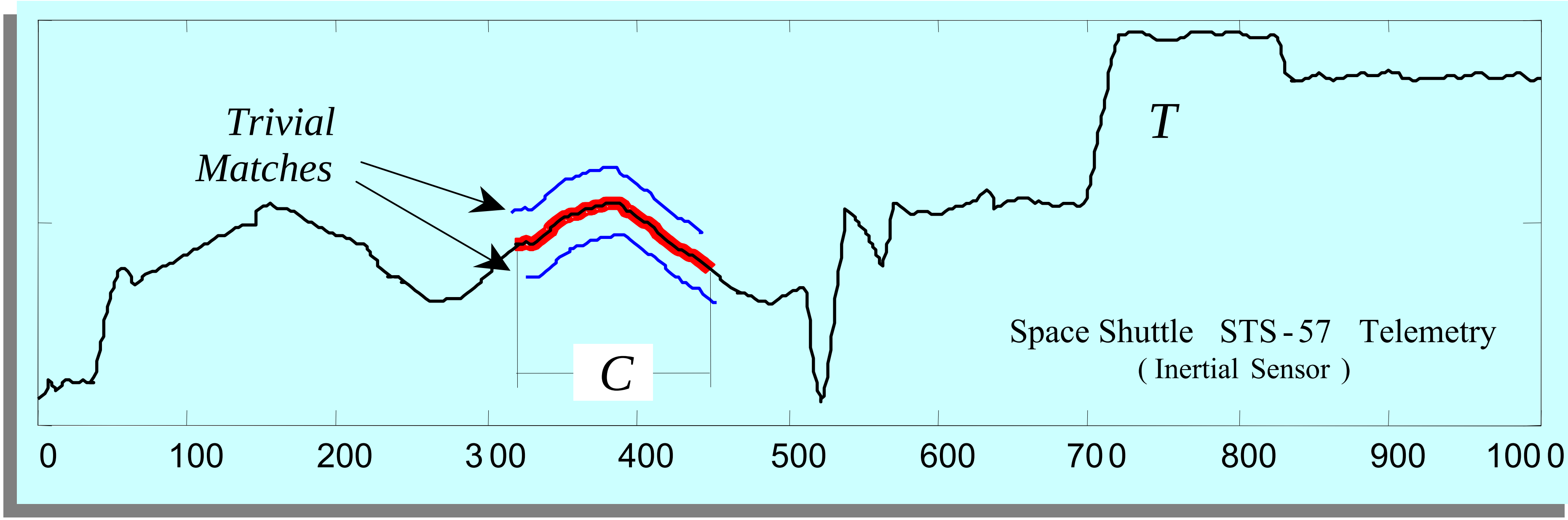
· Mining **association rules** in time series requires the discovery of motifs. These are referred to as *primitive shapes* and *frequent patterns*.

· Several time series **classification algorithms** work by constructing typical prototypes of each class. These prototypes may be considered motifs.

· Many time series **anomaly/interestingness detection** algorithms essentially consist of modeling normal behavior with a set of typical shapes (which we see as motifs), and detecting future patterns that are dissimilar to all typical shapes.

· In **robotics**, Oates et al., have introduced a method to allow an autonomous agent to generalize from a set of qualitatively different *experiences* gleaned from sensors. We see these "*experiences*" as motifs. See also Murakami Yoshikazu, Doki & Okuma and Maja J Mataric

· In **medical data mining**, Caraca-Valente and Lopez-Chavarrias have introduced a method for characterizing a physiotherapy patient's recovery based of the discovery of *similar patterns*. Once again, we see these "*similar patterns*" as motifs.



Definition 1. *Match*: Given a positive real number R (called *range*) and a time series T containing a subsequence C beginning at position p and a subsequence M beginning at q, if $D(C, M) \leq R$, then M is called a *matching* subsequence of C.

Definition 2. *Trivial Match*: Given a time series T, containing a subsequence C beginning at position p and a matching subsequence M beginning at q, we say that M is a *trivial match* to C if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.

Definition 3. *K-Motif*(n,R): Given a time series T, a subsequence length n and a range R, the most significant motif in T (hereafter called the *1-Motif*(n,R)) is the subsequence C_1 that has highest count of non-trivial matches (ties are broken by choosing the motif whose matches have the lower variance). The K^{th} most significant motif in T (hereafter called the *K-Motif*(n,R)) is the subsequence C_K that has the highest count of non-trivial matches, and satisfies $D(C_K, C_i) > 2R$, for all $1 \leq i < K$.

OK, we can define motifs, but how do we find them?

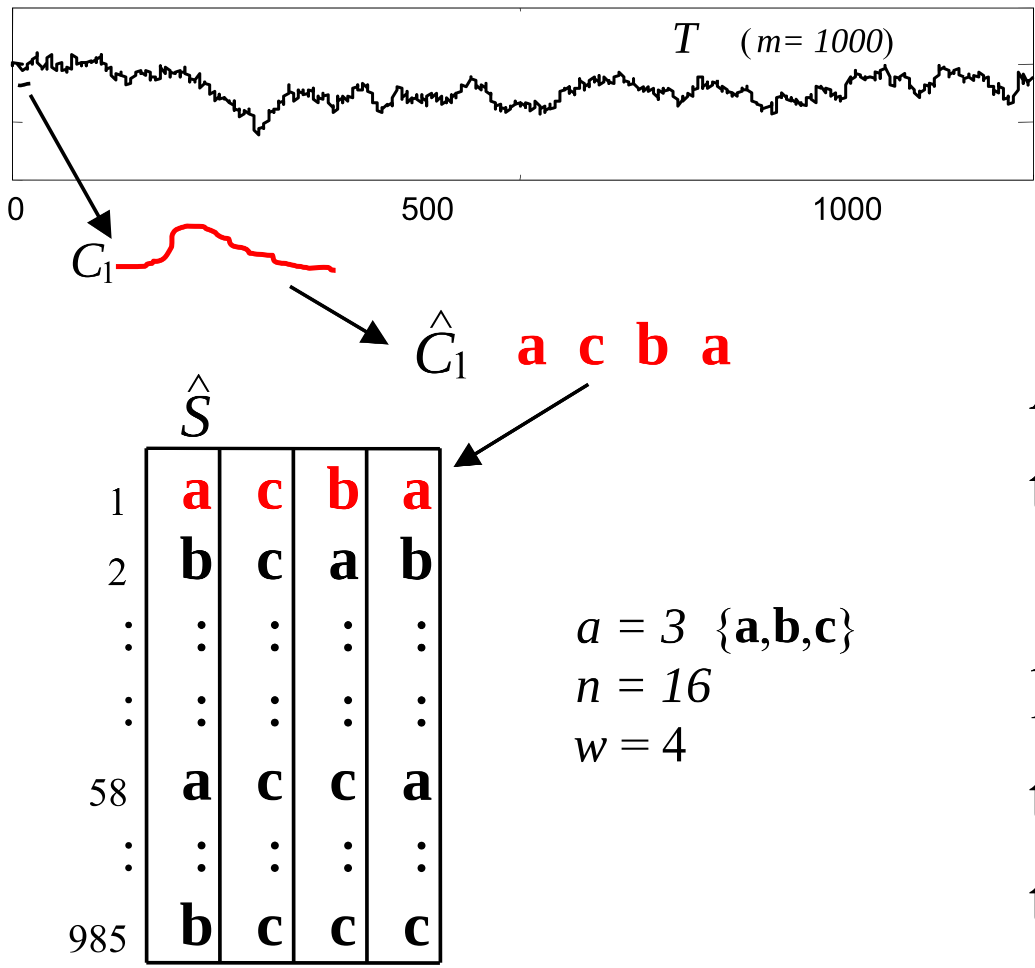
The obvious brute force search algorithm is just too slow…

The most reference algorithm is based on a *hot* idea from bioinformatics, *random projection** and the fact that SAX allows use to **lower bound** discrete representations of time series.

*** J Buhler and M Tompa. *Finding motifs using random projections*. In RECOMB'01. 2001.**

A simple worked example of the motif discovery algorithm

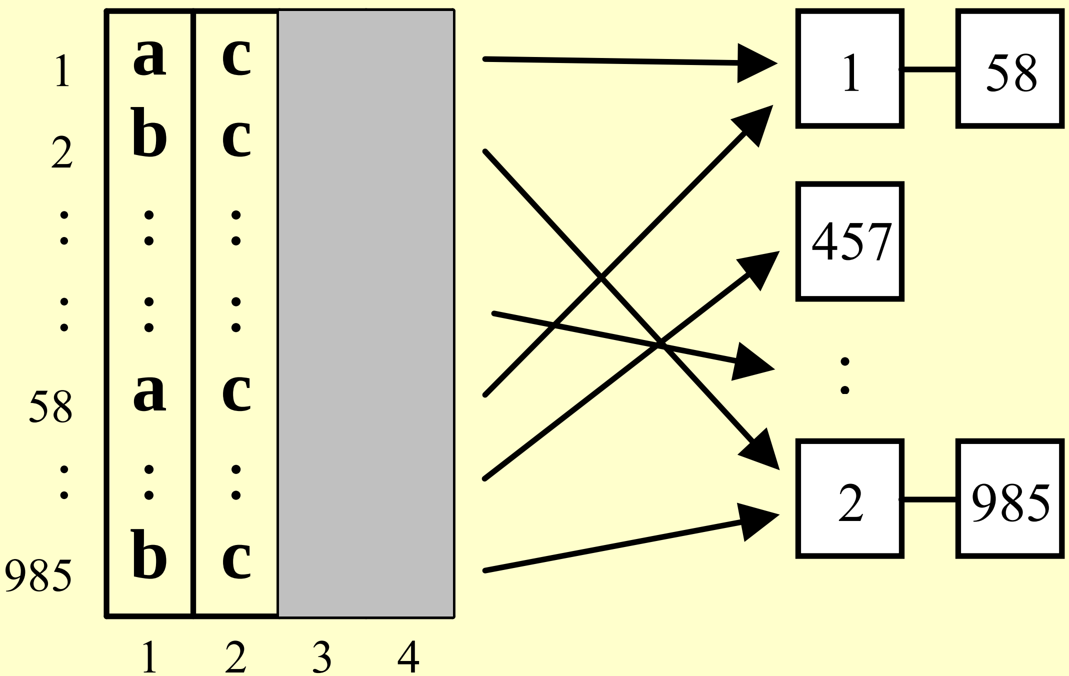
The next 4 slides



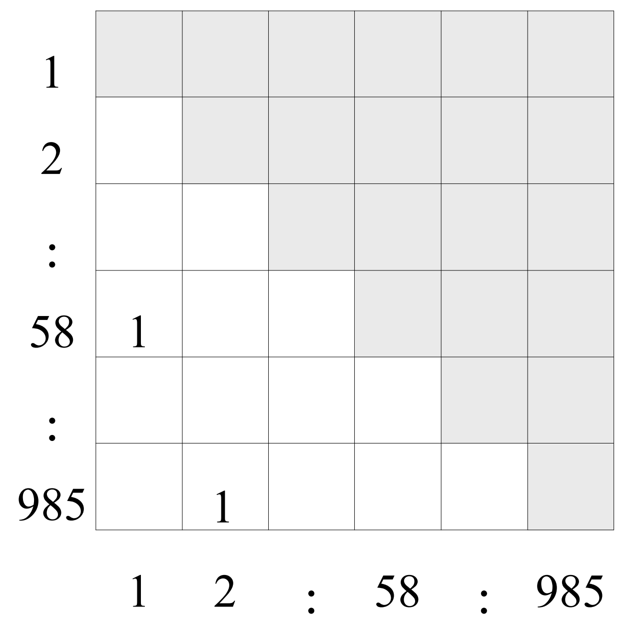
$a = 3$ {**a**,**b**,**c**}
$n = 16$
$w = 4$

Assume that we have a time series T of length 1,000, and a motif of length 16, which occurs twice, at time T_1 and time T_{58}.

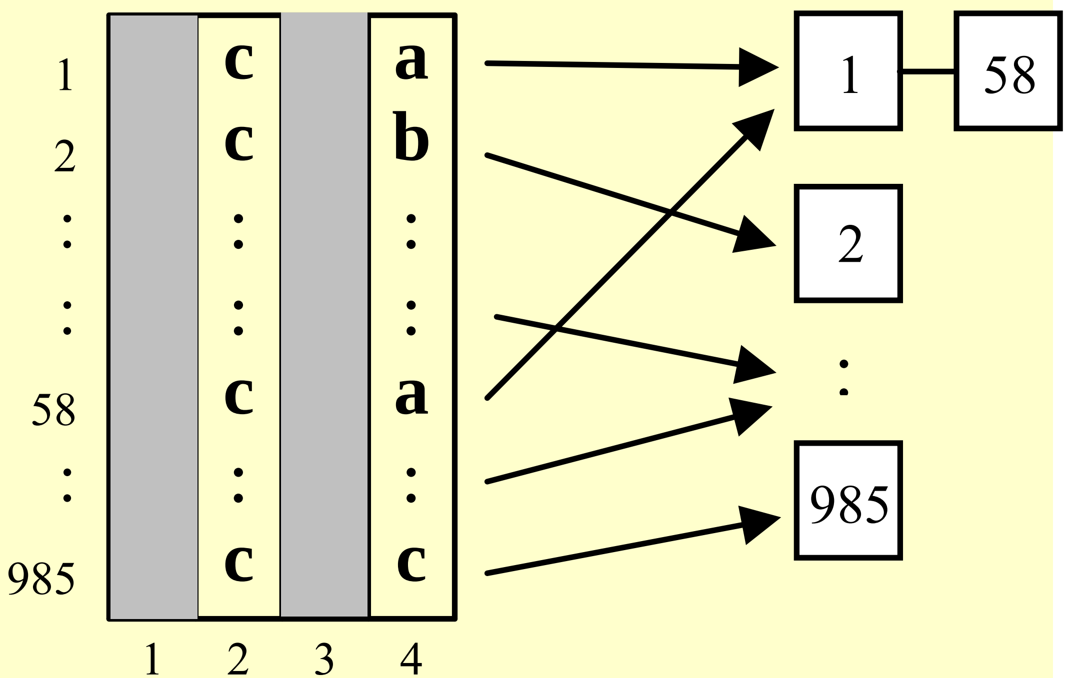
A mask {1,2} was randomly chosen, so the values in columns {1,2} were used to project matrix into buckets.



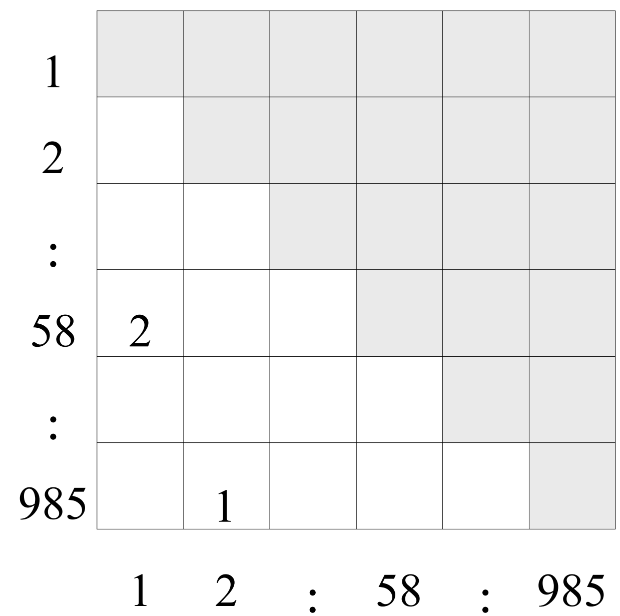
Collisions are recorded by incrementing the appropriate location in the collision matrix



A mask {2,4} was randomly chosen, so the values in columns {2,4} were used to project matrix into buckets.



Once again, collisions are recorded by incrementing the appropriate location in the collision matrix



We can now use the information in the collision matrix as a heuristic to hunt for likely motifs.

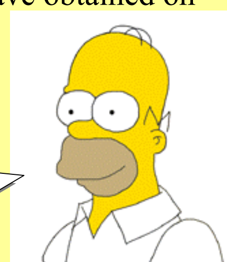
We can use lower bounding to discover at what point that hunt is fruitless…

This is a good example of the Generic Data Mining Algorithm…

The Generic Data Mining Algorithm

- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest
- Approximately solve the problem at hand in main memory
- Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

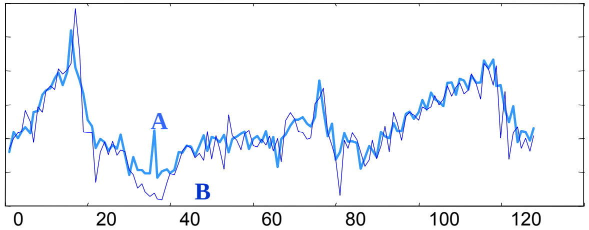
But which *approximation* should we use?

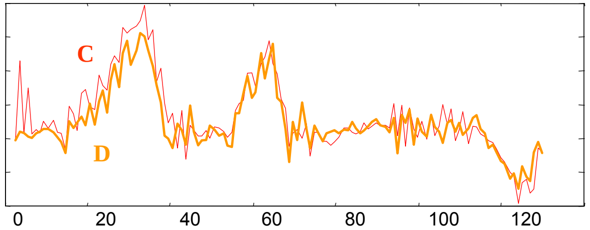


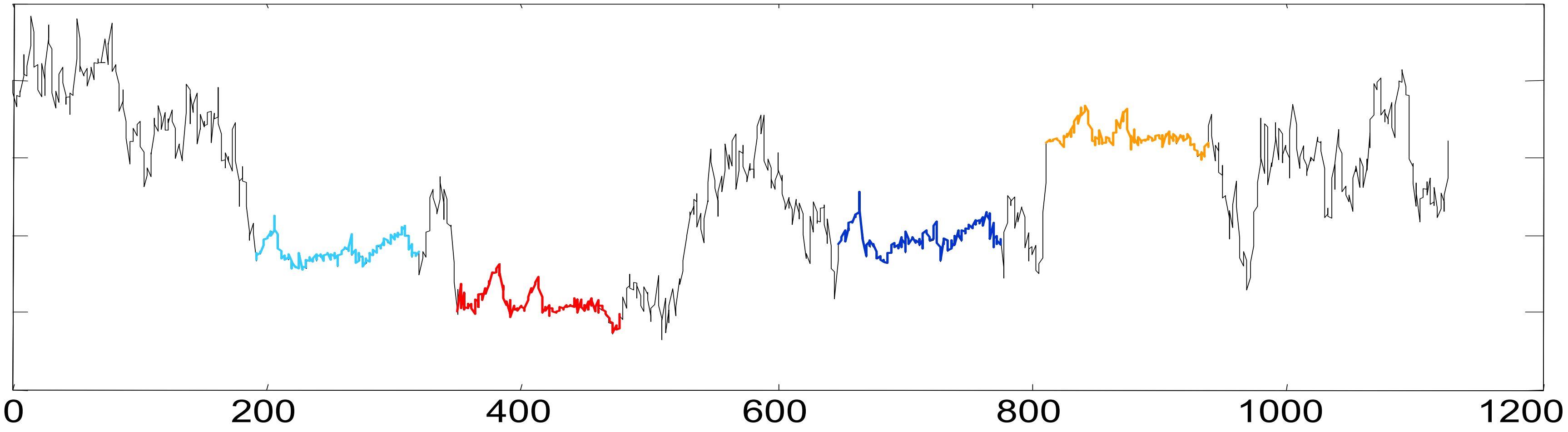
1						
2	2					
:						
58	**27**					
:	3			1		
985		2		1		
	1	2	:	58	:	985

A Simple Experiment

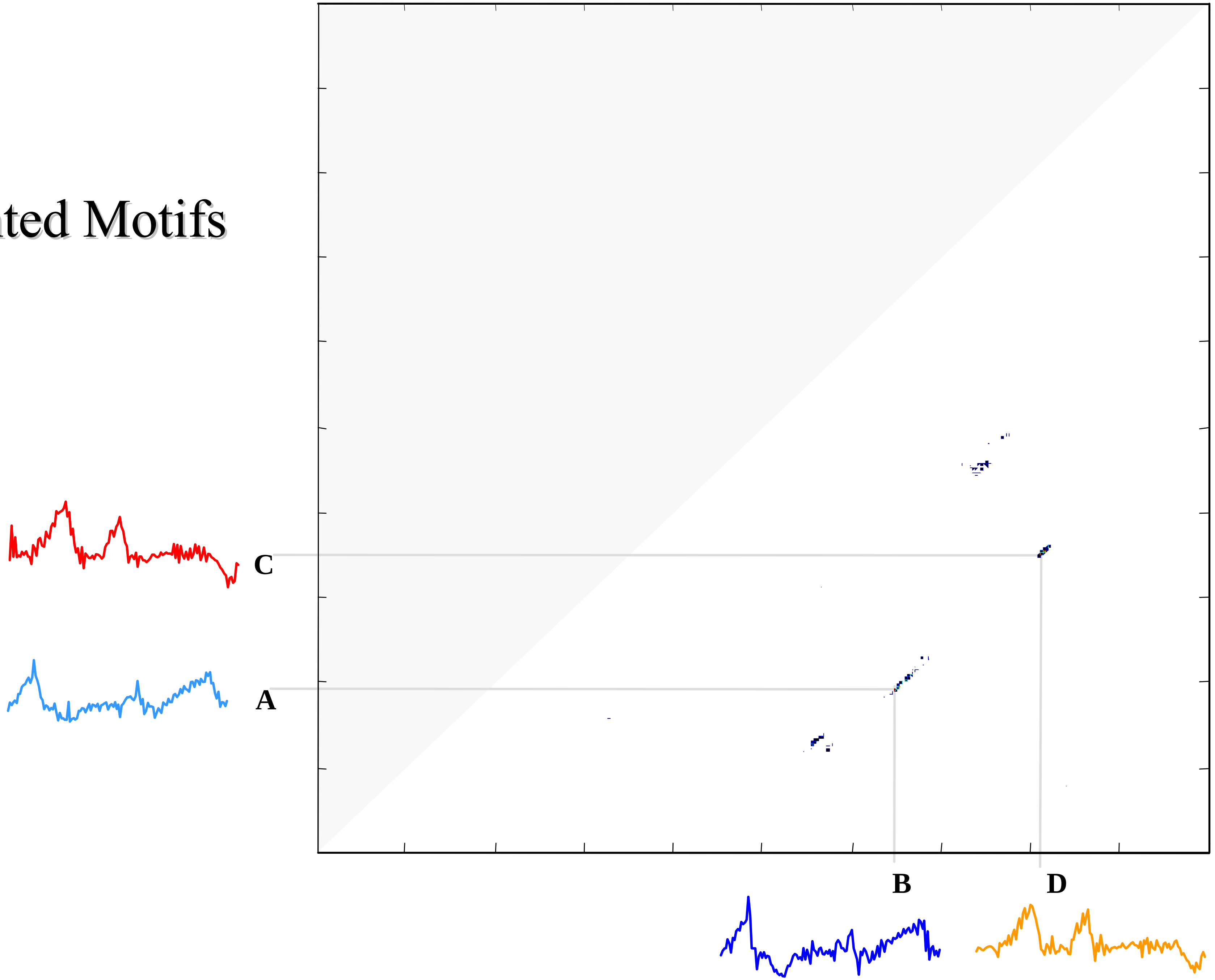
Let us imbed two motifs into a random walk time series, and see if we can recover them







Planted Motifs

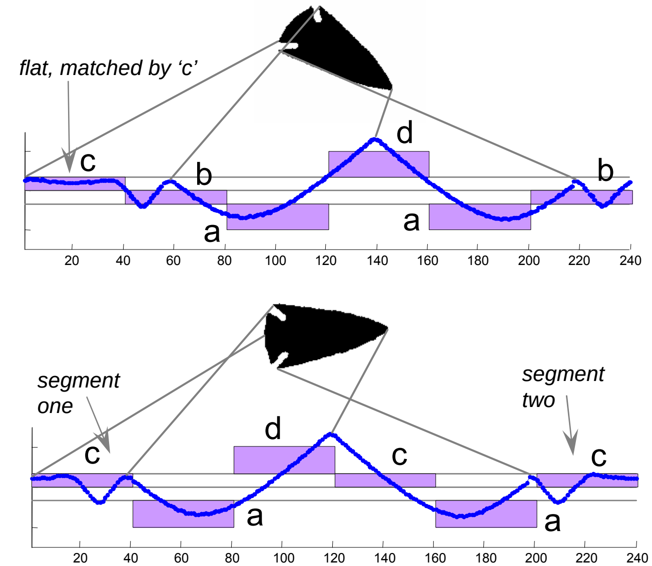


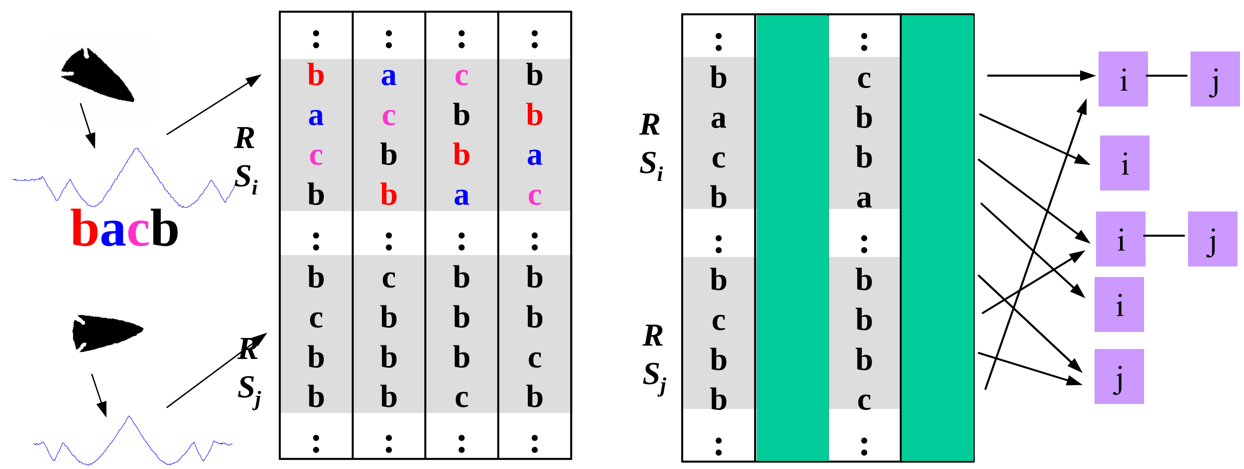
Shape Motifs I

We can find shape motifs with only minor modifications:

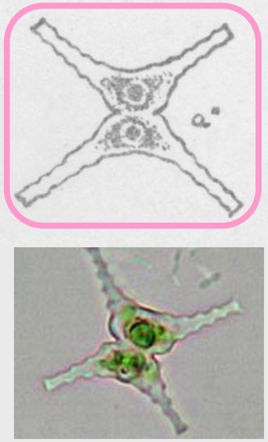
• When converting shape to SAX, try all rotations to fit best fit.

• Place every circular shift of SAX word in the projection matrix.

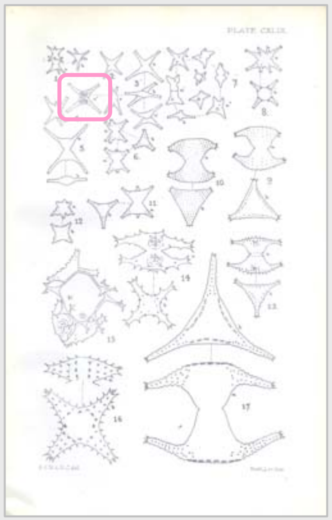


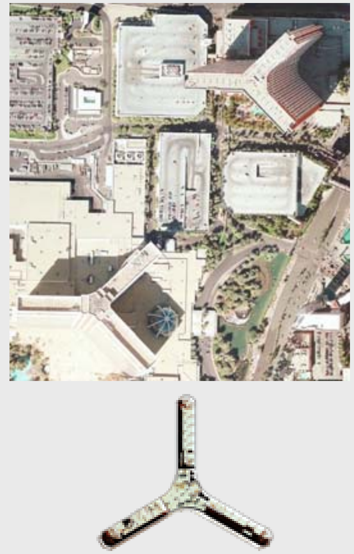


Shape Motifs II

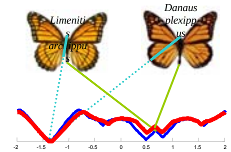


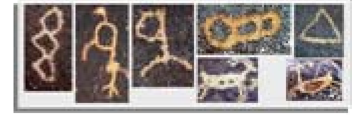
Staurastrum tetracerum



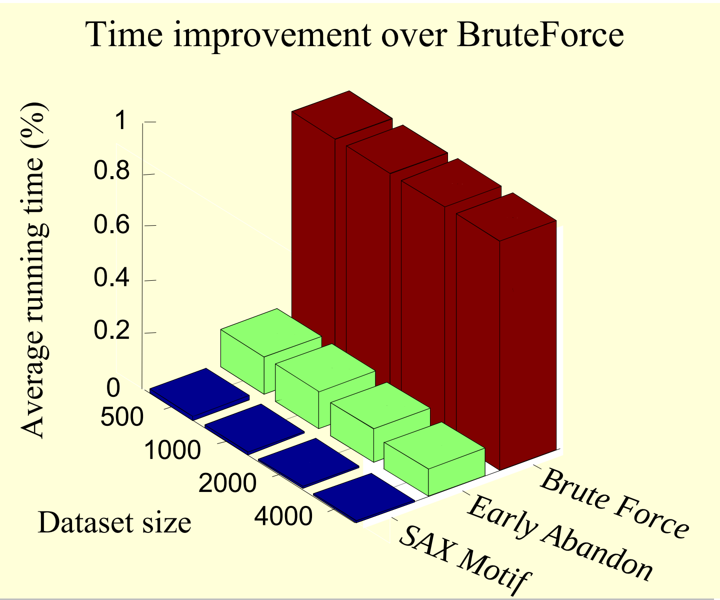


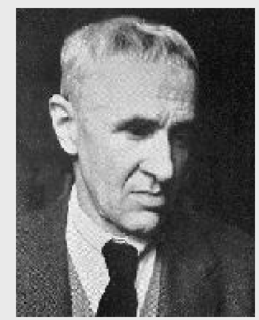
“Vegas” Motif





graffiti





Giorgio Morandi

1890 –1964

Through his simple and repetitive ***motifs*** *… Morandi became an important forerunner of Minimalism.*

wikipedia

Image Discords

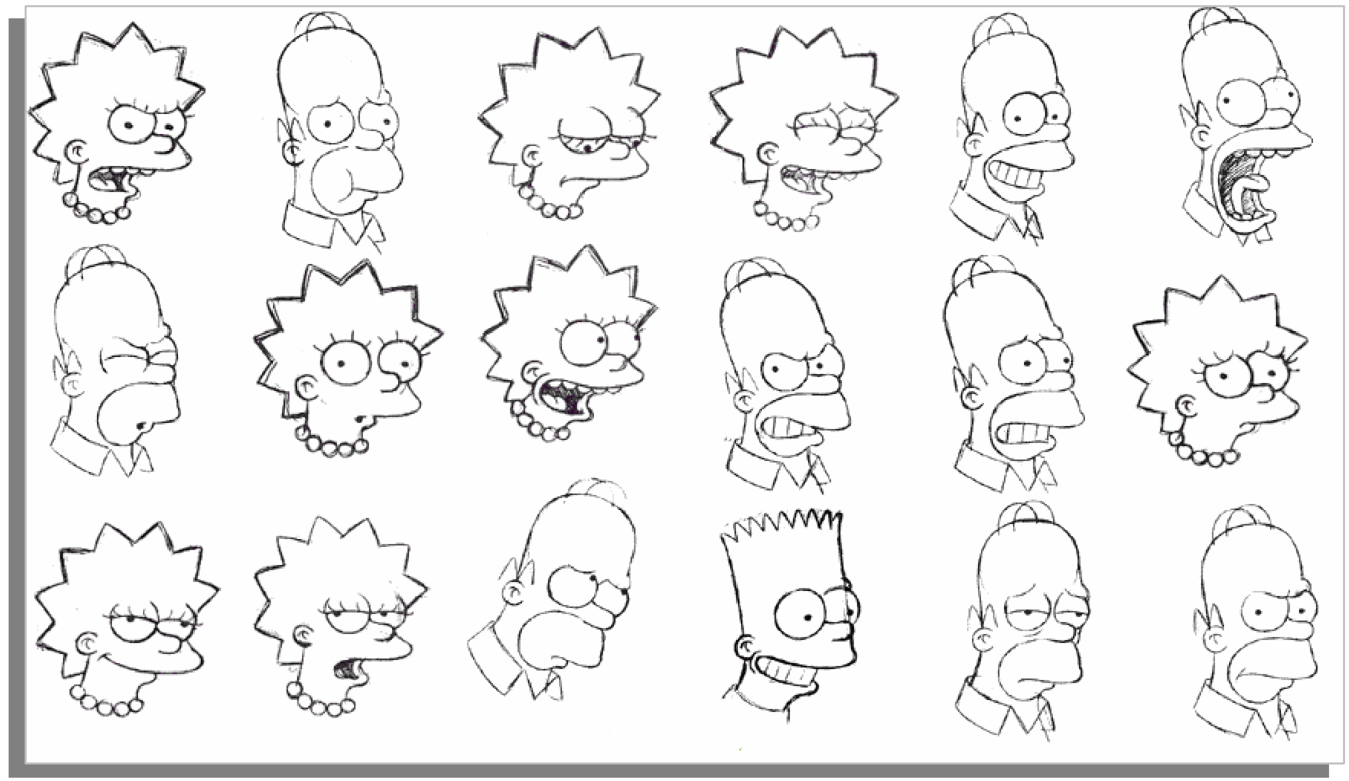
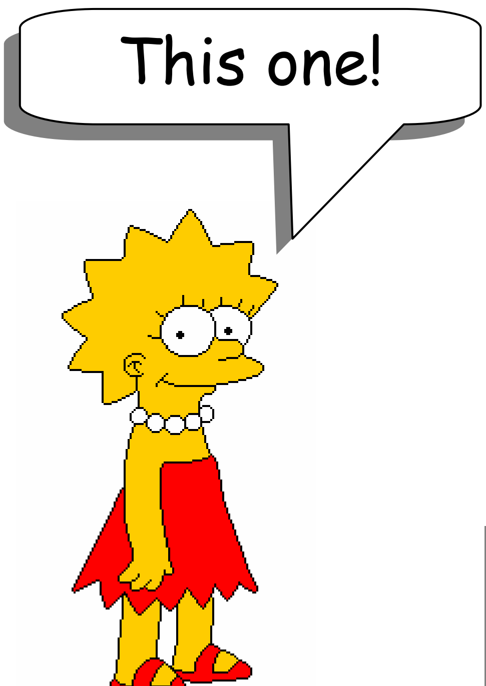
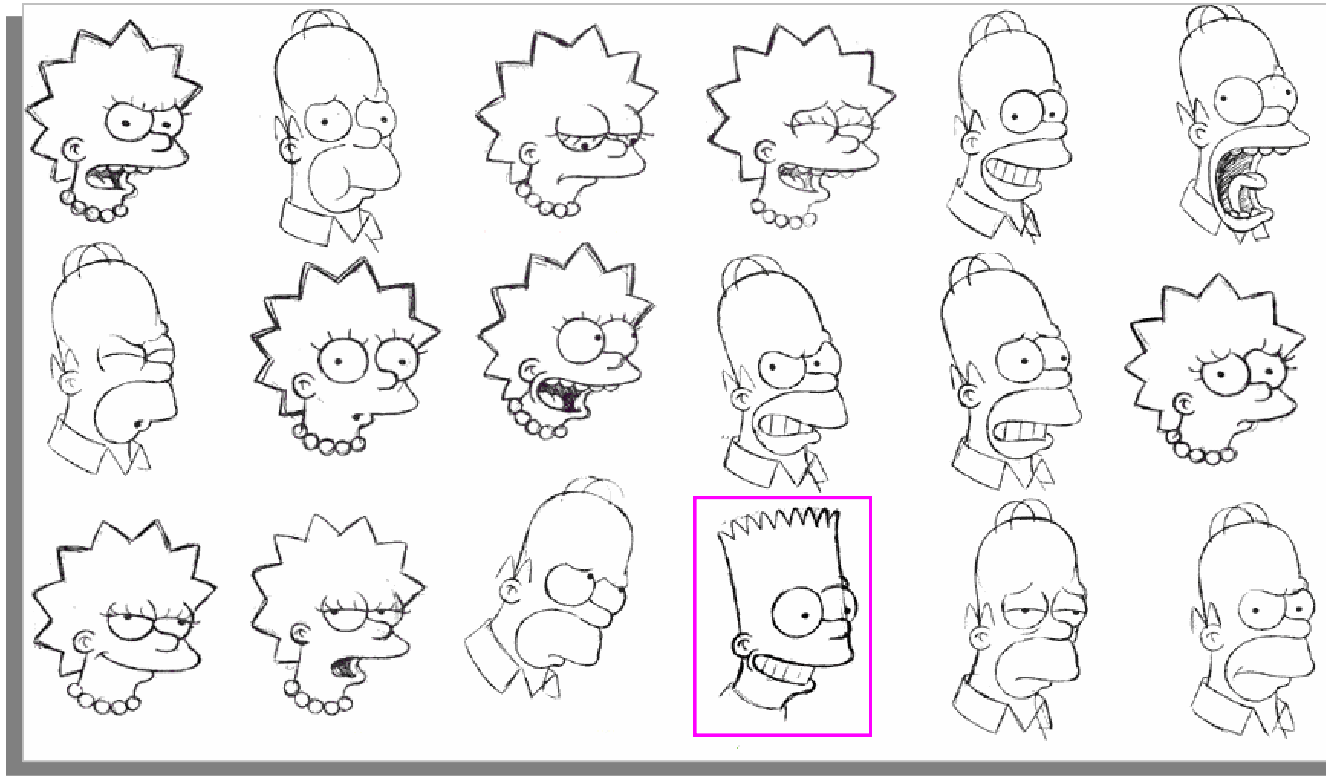


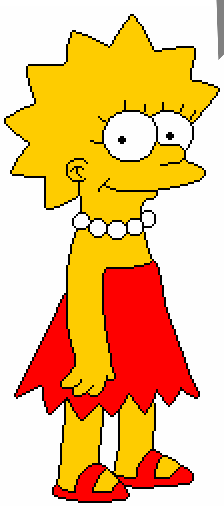
Image Discords

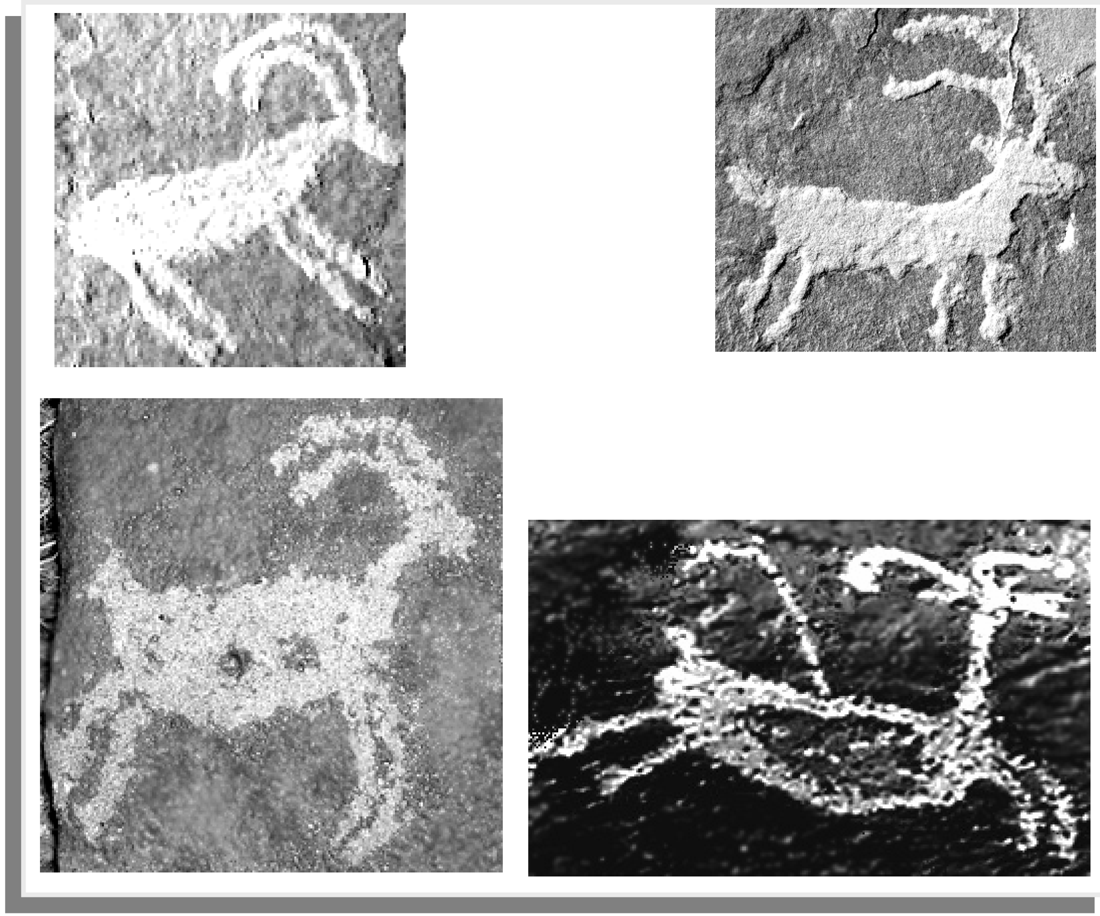


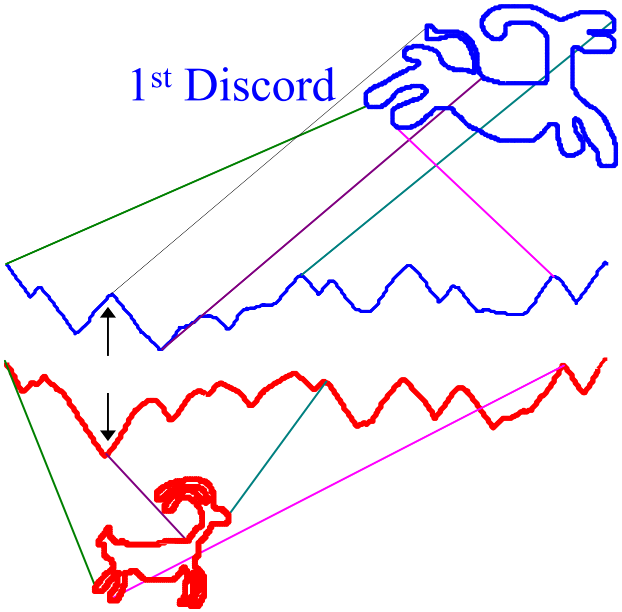


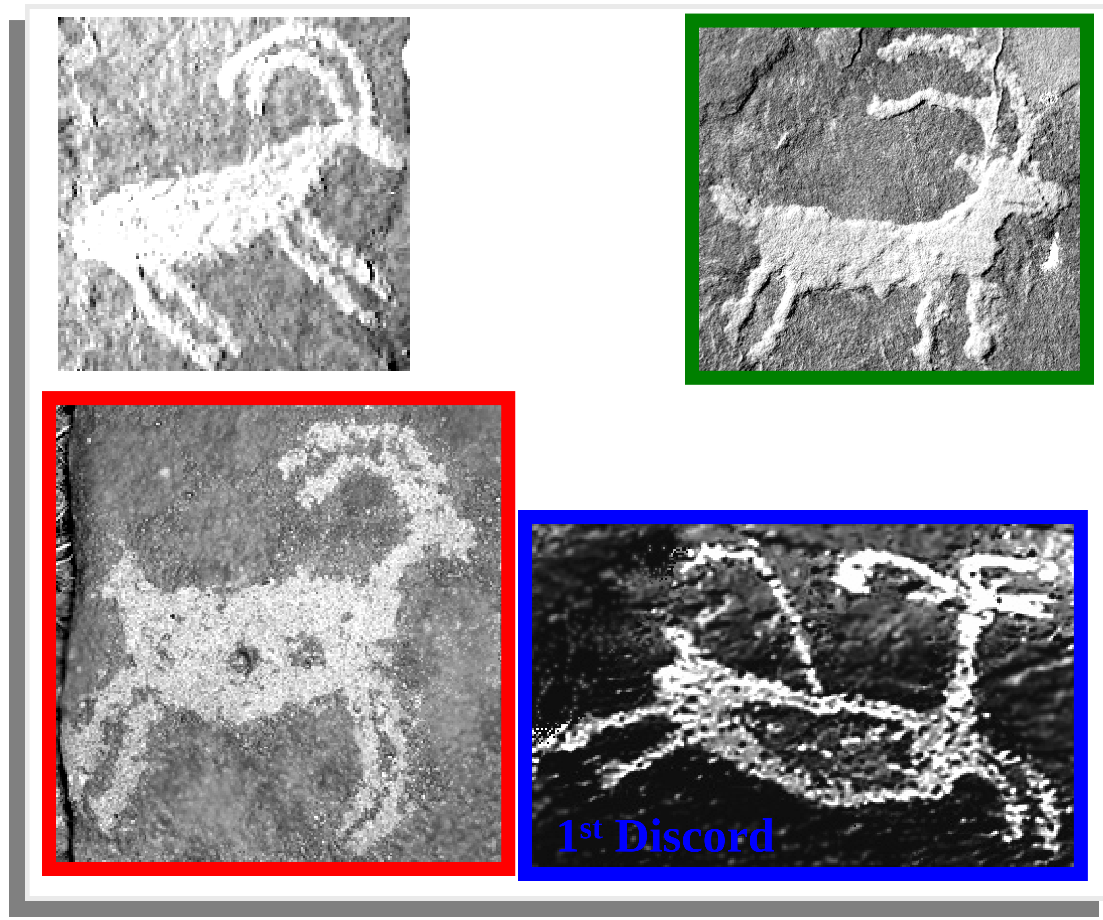
Shape Discord: Given a collection of shapes S, the shape D is the discord of S if D has the largest distance to its nearest match. That is, $\forall$ shape C in S, the nearest match M_C of C and the nearest match M_D of D, $Dist(D, M_D) > Dist(C, M_C)$.

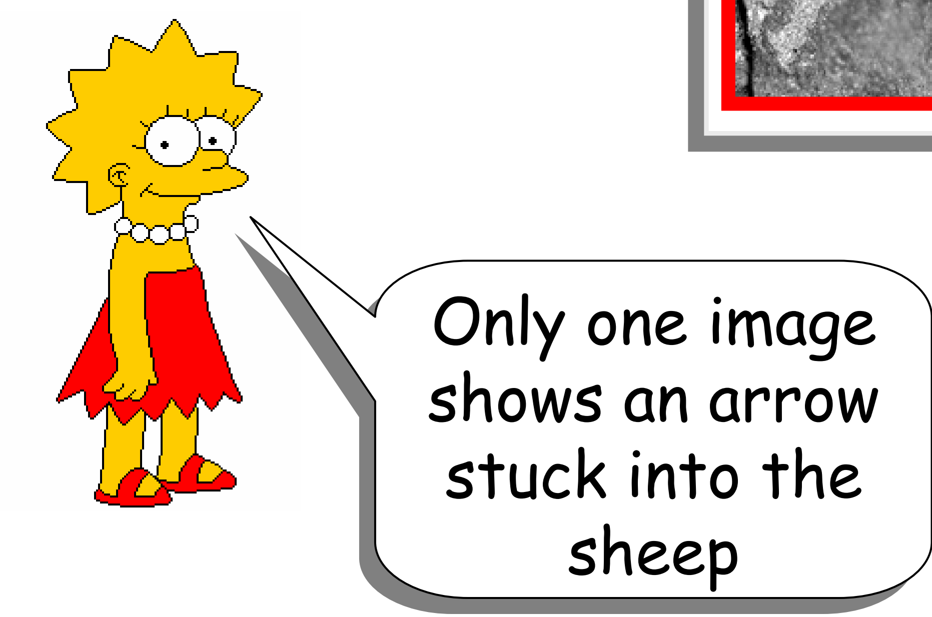
This one is even more subtle… Here is a subset of a large collection of petroglyphs






1st Discord


1st Discord


Only one image shows an arrow stuck into the sheep

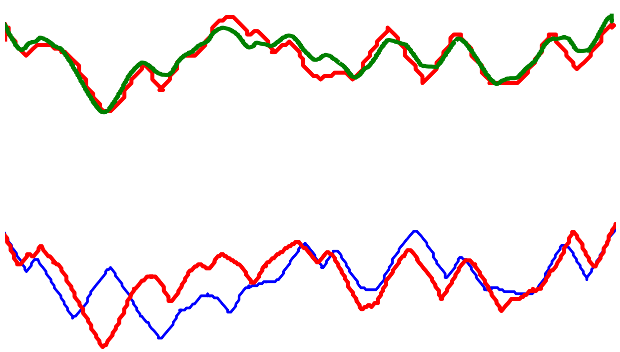
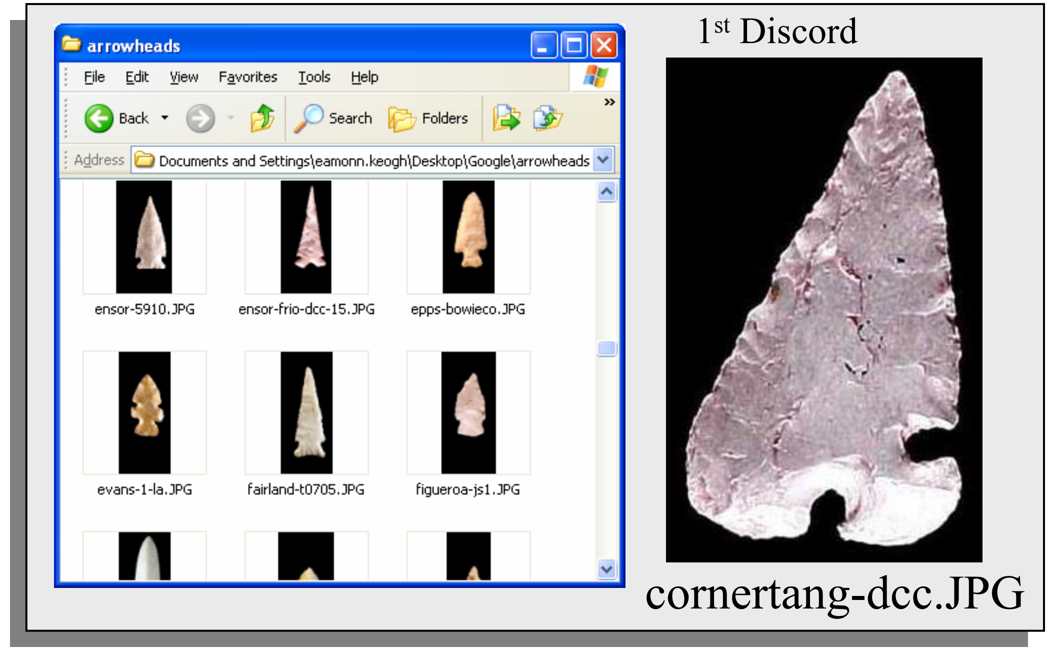
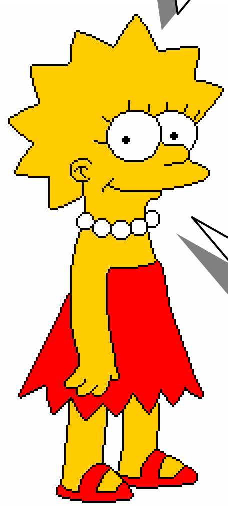
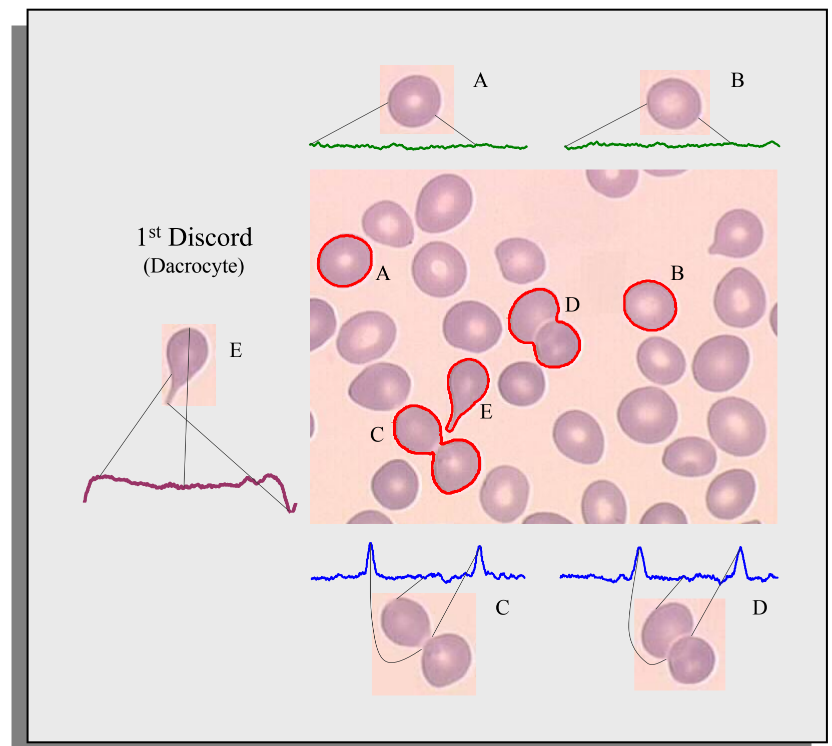


Image discords are potentially useful in many domains…
Most arrowheads are symmetric, but…





Most red blood cells are round…



Finding Image Discords

0	**2**	4.2	1.1	2.3	8.5
2	0	3	3.2	3.5	8.2
4.2	3	0	1.2	9.2	9.7
1.1	3.2	**1.2**	0	**0.1**	**7.5**
2.3	3.5	9.2	**0.1**	0	7.6
8.5	8.8	9.7	7.5	7.6	0

1.1	2	1.2	0.1	0.1	7.5

```
Function  [ dist, loc ] = Discord_Search(S)
best_so_far_dist = 0
best_so_far_loc = NaN
for p  = 1 to size (S)                          // begin outer loop
  nearest_neighbor_dist = infinity
  for q  = 1 to size (S)                        // begin inner loop
    if p!= q                                    // Don't compare to self
        if  RD(Cp , Cq)  < nearest_neighbor_dist
          nearest_neighbor_dist = RD(Cp , Cq)
        end
    end
  end                                           // end inner loop
   if nearest_neighbor_dist > best_so_far_dist
     best_so_far_dist = nearest_neighbor_dist
     best_so_far_loc  = p
   end
end                                             // end outer loop
return [ best_so_far_dist, best_so_far_loc ]
```

Finding Discords, Fast

```
Function [ dist, loc ] = Heuristic_Search(S, Outer, Inner )
best_so_far_dist = 0
best_so_far_loc = NaN
for each index p given by heuristic Outer   // begin outer loop
  nearest_neighbor_dist = infinity
  for each index q given by heuristic Inner // begin inner loop
    if p!= q
      if RD(C_p, C_q) < best_so_far_dist
        break                                // break out of inner loop
      end
      if RD(C_p, C_q) < nearest_neighbor_dist
        nearest_neighbor_dist = RD(C_p, C_q)
       end
    end
  end                                        // end inner loop
   if nearest_neighbor_dist > best_so_far_dist
     best_so_far_dist = nearest_neighbor_dist
     best_so_far_loc  = p
   end
end                                          // end outer loop
return [ best_so_far_dist, best_so_far_loc ]
```

0	**2**	4.2	1.1	2.3	8.5
2	0	3	3.2	3.5	8.2
4.2	3	0	1.2	9.2	9.7
1.1	3.2	**1.2**	0	**0.1**	**7.5**
2.3	3.5	9.2	**0.1**	0	7.6
8.5	8.8	9.7	7.5	7.6	0

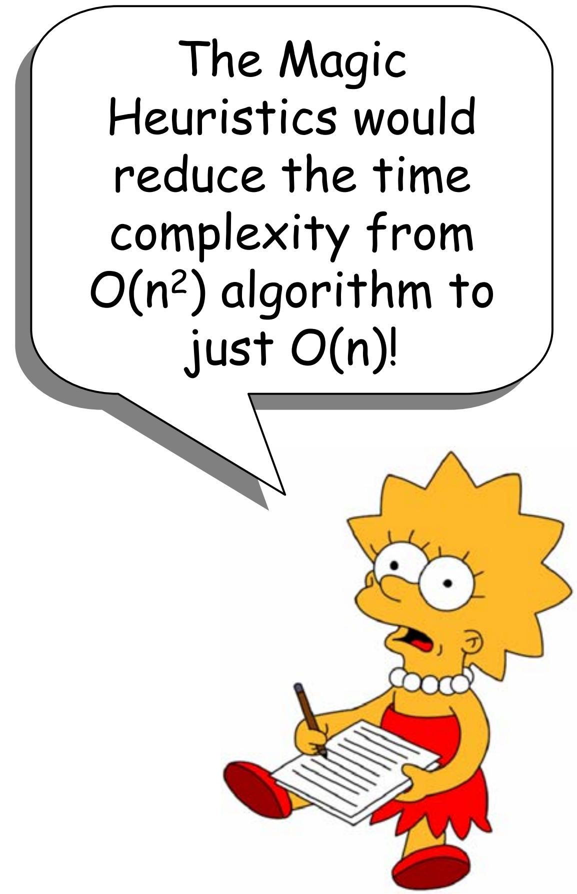
The code now says…
If while searching a given column, you find a distance less than nearest_neighbor_dist then that column cannot have the discord.

The code also uses heuristics to order the search…

The Magic Heuristics

• In the outer loop, visit the columns in order of the Discord score
• In the inner loop, visit the row cells in order of nearest neighbor first

0	**2**	4.2	1.1	2.3	8.5
2	0	3	3.2	3.5	8.2
4.2	3	0	1.2	9.2	9.7
1.1	3.2	**1.2**	0	**0.1**	**7.5**
2.3	3.5	9.2	**0.1**	0	7.6
8.5	8.8	9.7	7.5	7.6	0



The Magic Heuristics

0	2	4.2	1.1	2.3	8.5
2	0	3	3.2	3.5	8.2
4.2	3	0	1.2	9.2	9.7
1.1	3.2	1.2	0	0.1	7.5
2.3	3.5	9.2	0.1	0	7.6
8.5	8.8	9.7	7.5	7.6	0

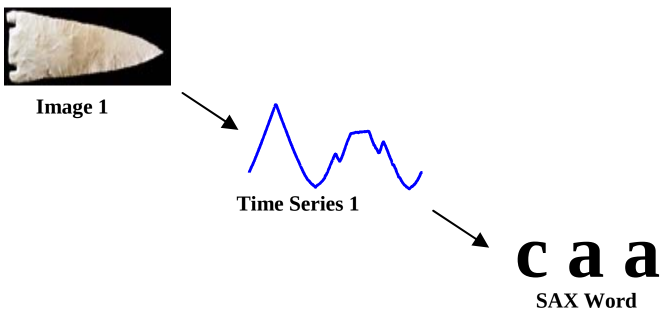
- In the outer loop, visit the columns in order of the Discord score
- In the inner loop, visit the row cells in order of nearest neighbor first

Observations

- Visiting the columns in *approximately* order of the Discord score is still very helpful
- For the inner loop, we don't really need visit the rows in order of nearest neighbor first, so long as we find a "*near enough*" neighbor early on

Approximately Magic Heuristics

0	2	4.2	1.1	2.3	8.5
2	0	3	3.2	3.5	8.2
4.2	3	0	1.2	9.2	9.7
1.1	3.2	1.2	0	0.1	7.5
2.3	3.5	9.2	0.1	0	7.6
8.5	8.8	9.7	7.5	7.6	0

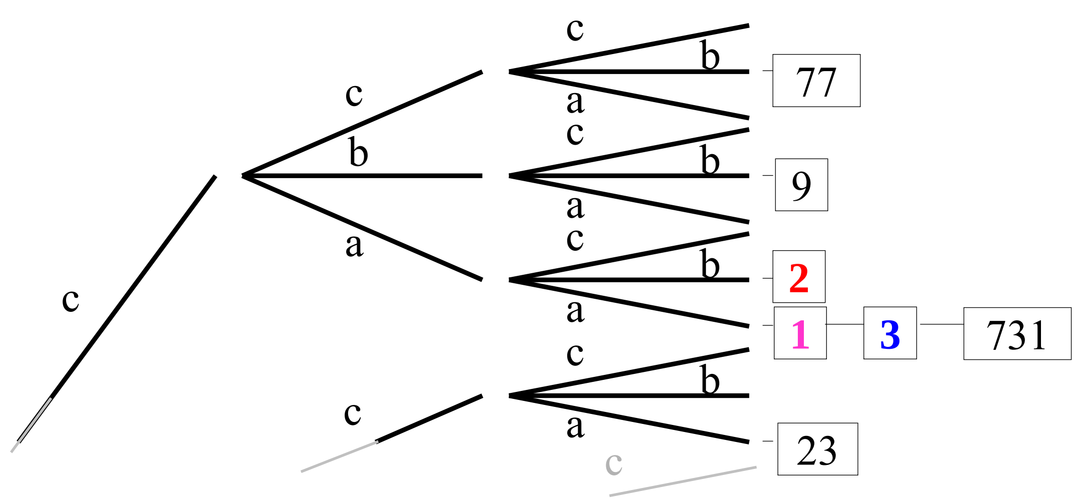


Rotation invariance ignored here

Inserted into array

1	c	a	a	3
2	c	a	b	1
3	c	a	a	3
::	::	::	::	::
::	::	::	::	::
	c	b	b	2
m-1	a	c	b	1
m	b	c	a	2

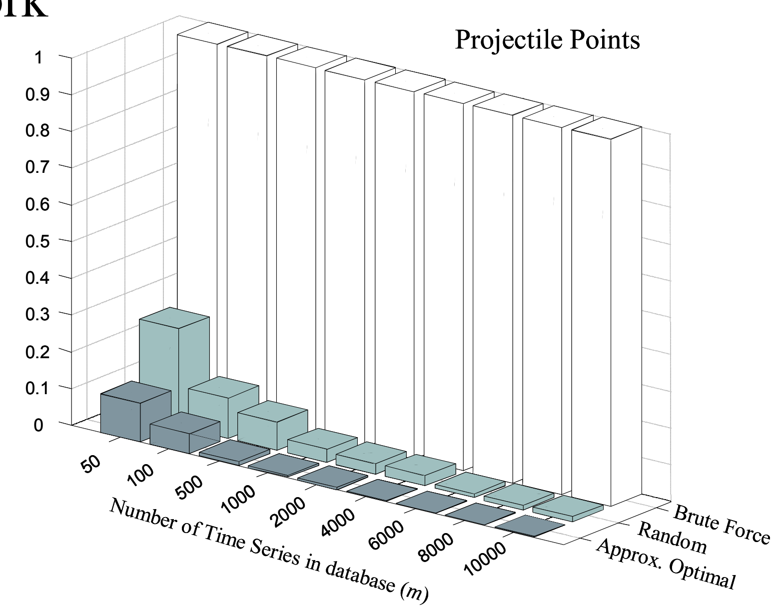
Augmented Trie



How Fast is Approximately Magic?

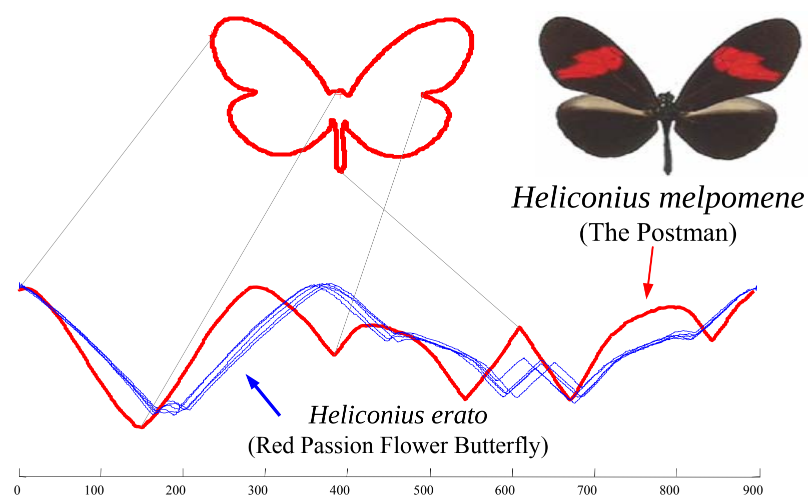
On a problem dataset of arrowheads

- If we only see 200 arrowheads, we do an extra 21.8% more work than the Magic algorithm
- For larger arrowhead datasets we get even closer to Magic algorithm
- In other words, we are doing O(n) work, not O(n^2) work.
- Empirically we see similar results for other datasets, but in pathological datasets, we can still be forced to do O(n^2) work

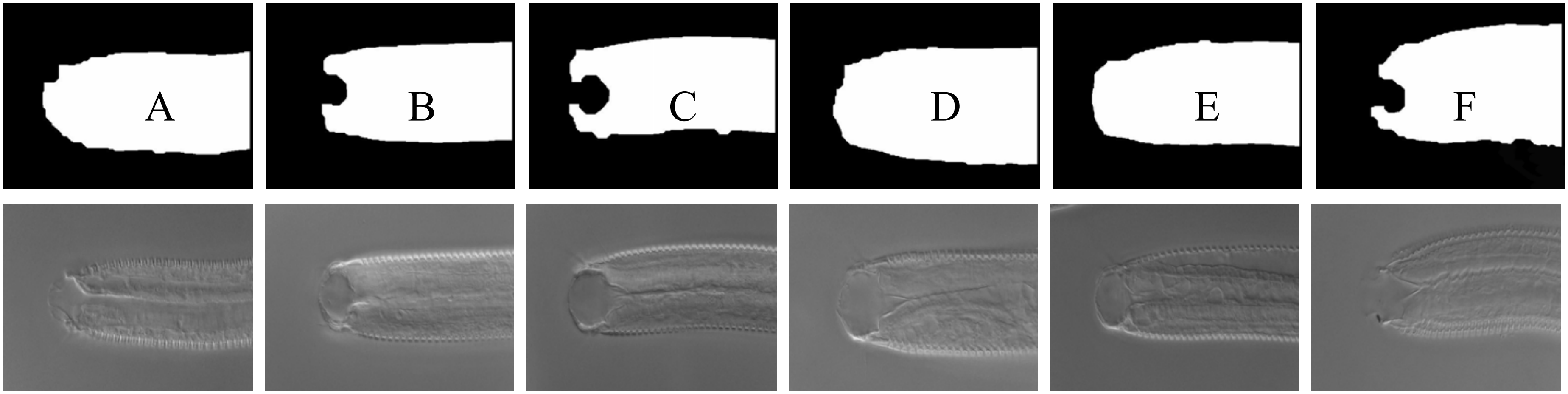


Which is the "odd man out" in this collection of Red Passion Flower Butterflies?

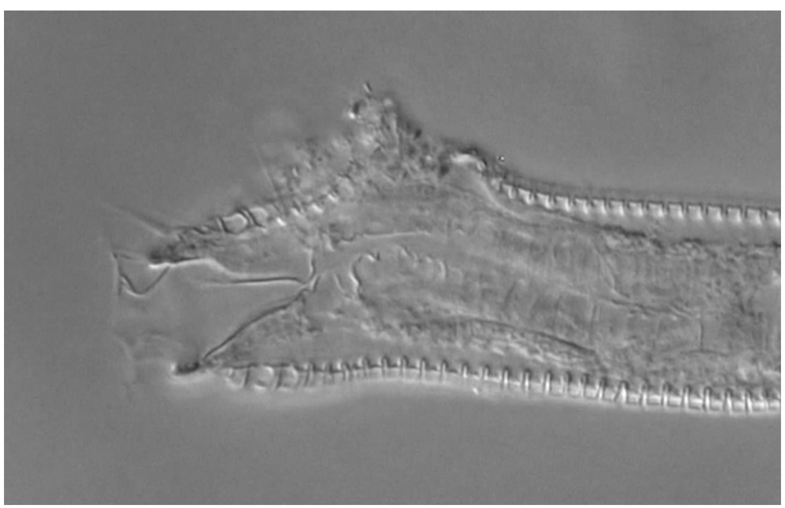
One of them is *not* a Red Passion Flower Butterfly. A fact that can be discovered by finding the shape discord

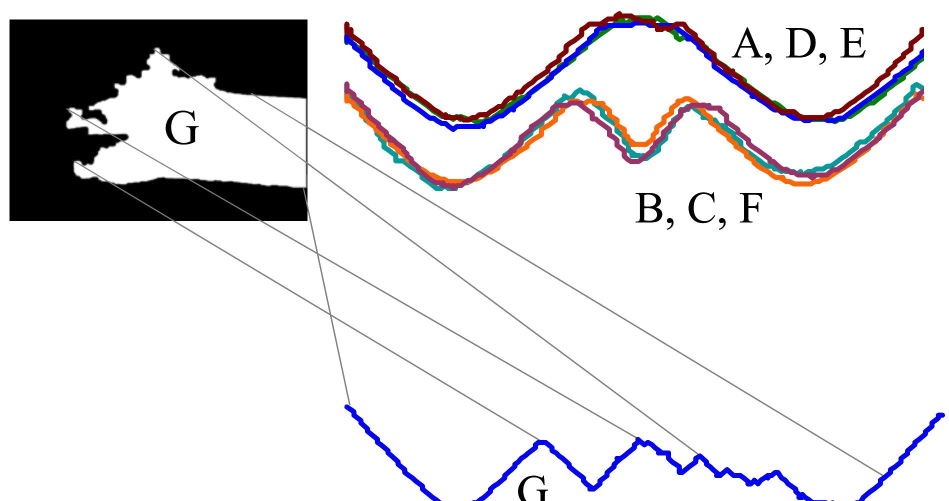


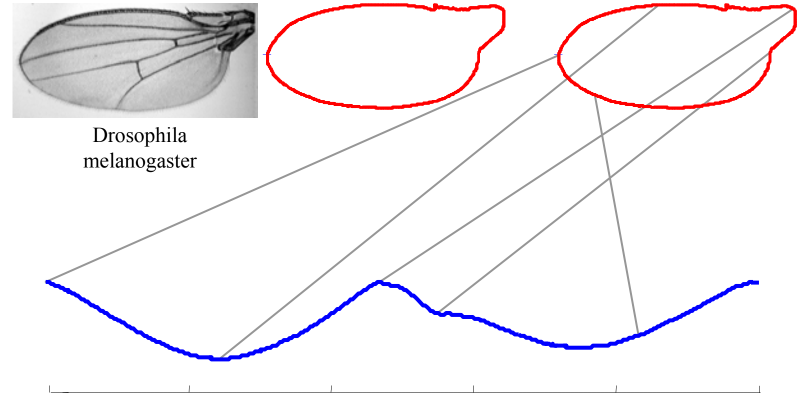
Nematode Discords

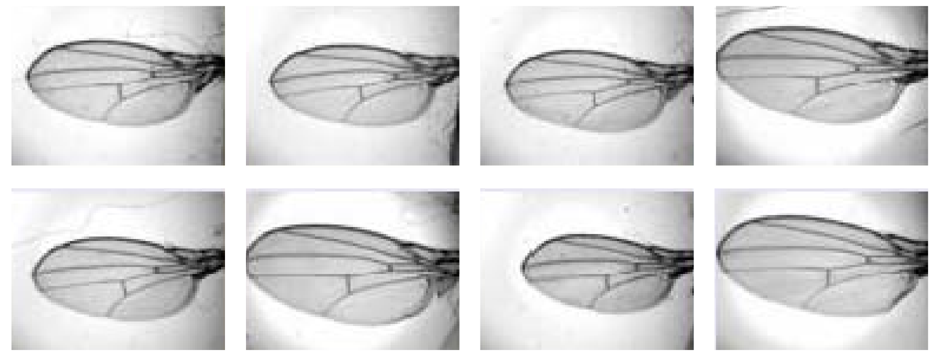


Though 20,000 species have been classified it is estimated that this number might be upwards of 500,000 if all were known. *Wikipedia*

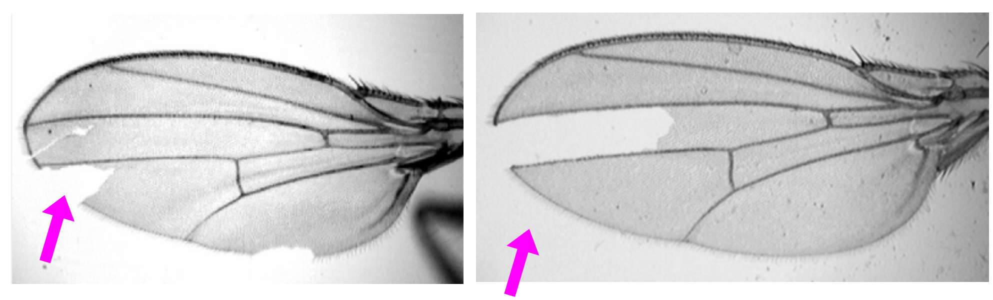


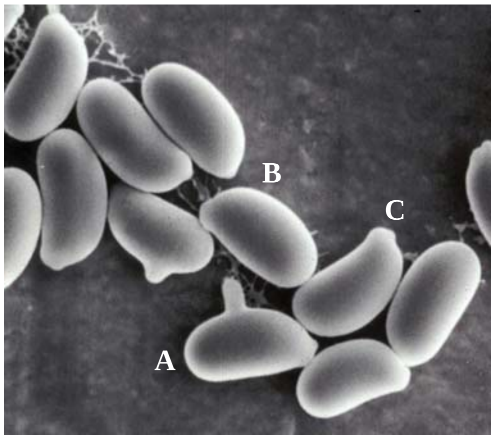


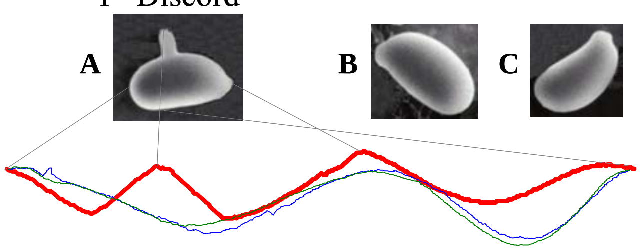




A subset of 32,028 images of Drosophila wings



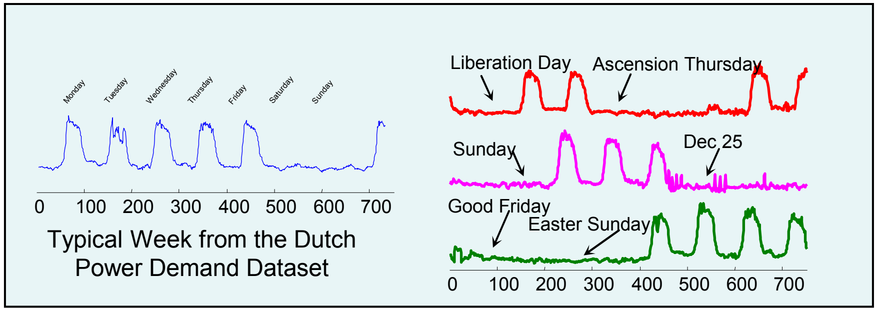


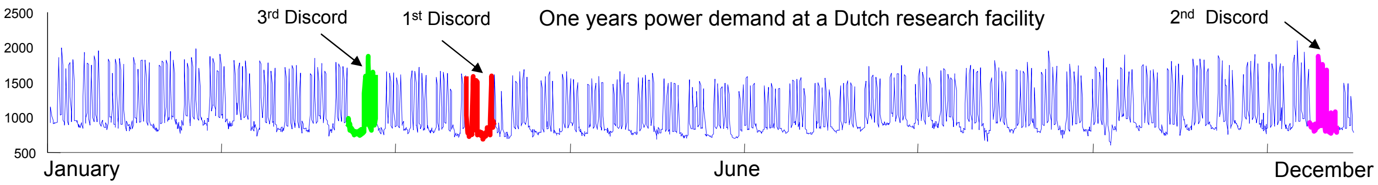


Fungus Images

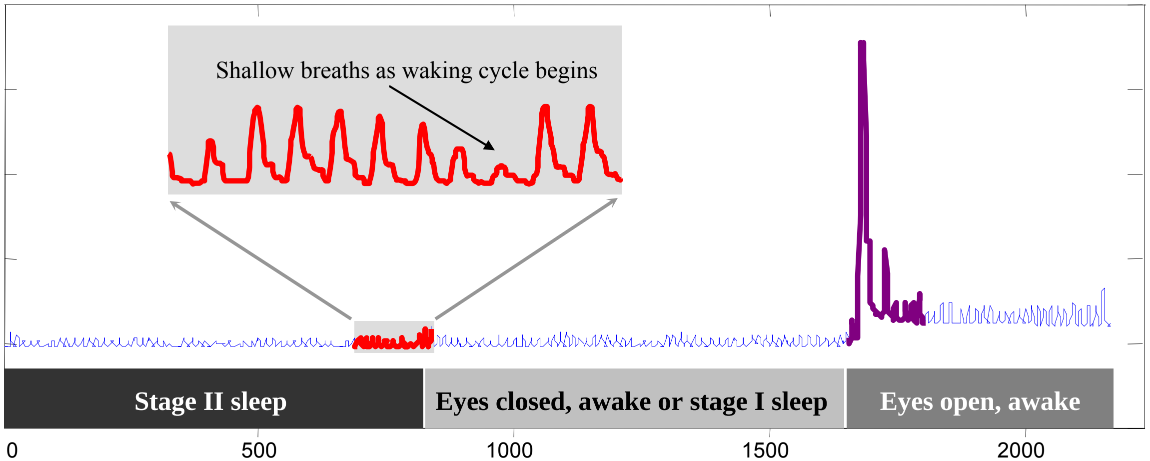
Some spores produced by a rust (fungus) known as Gymnosporangium, which is a parasite of apple and pear trees. Note that one spore has sprouted an “appendage” known as a germ tube, and is thus singled out as the discord.

Time Series Discords





Power Demand



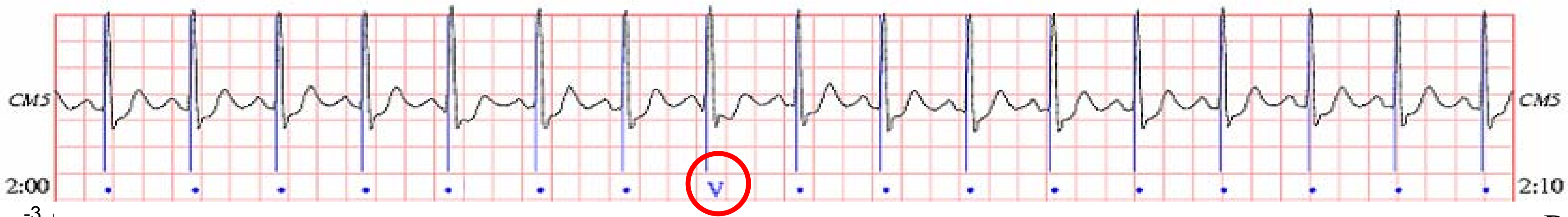
Sleep Cycles

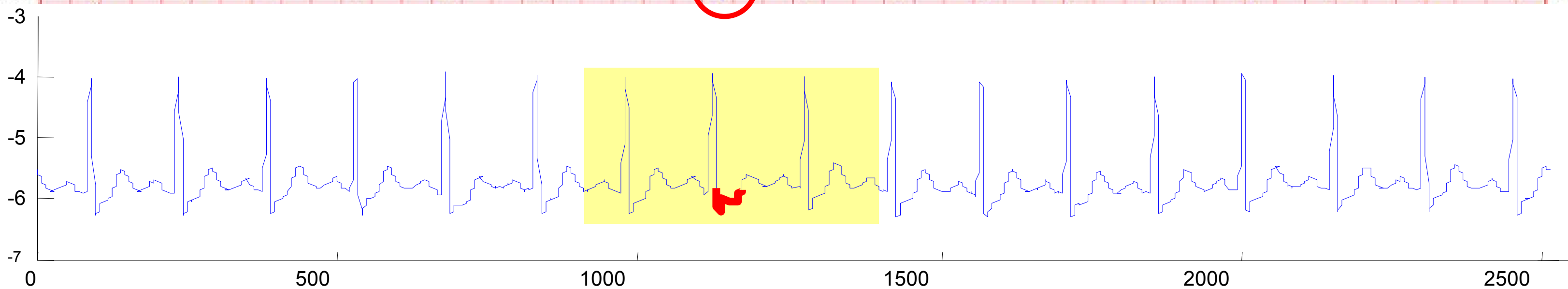
A time series showing a patients respiration (measured by thorax extension), as they wake up. A medical expert, Dr. J. Rittweger, manually segmented the data. The 1-discord is a very obvious deep breath taken as the patient opened their eyes. The 2-discord is much more subtle and impossible to see at this scale. A zoom-in suggests that Dr. J. Rittweger noticed a few shallow breaths that indicated the transition of sleeping stages.

Institute for Physiology. Free University of Berlin. Data shows respiration (thorax extension), sampling rate 10 Hz.

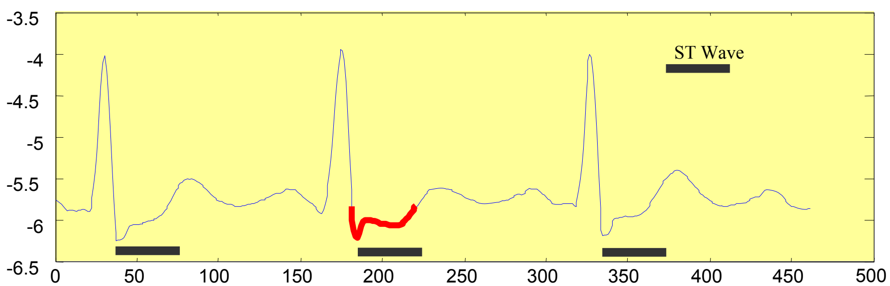
Discords in Medical Data

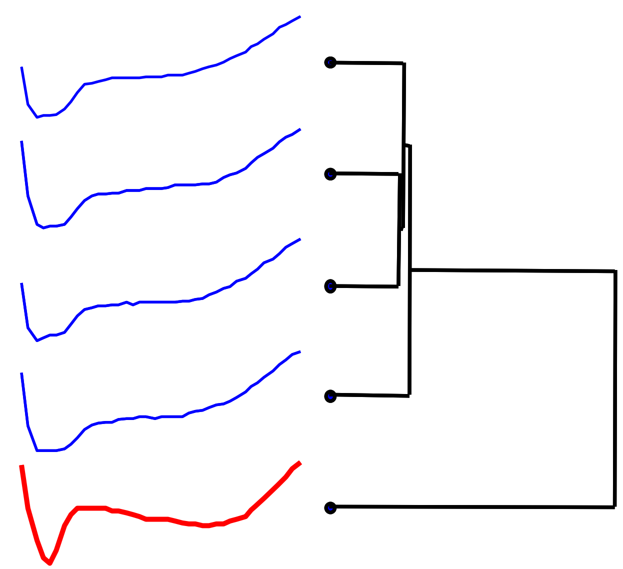
A cardiologist noted subtle anomalies in this dataset. Let us see if the discord algorithm can find them.



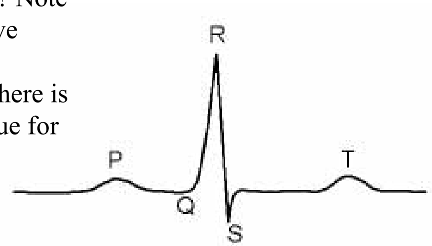


Record qtdbsele0606 from the PhysioBank QT Database (qtdb)



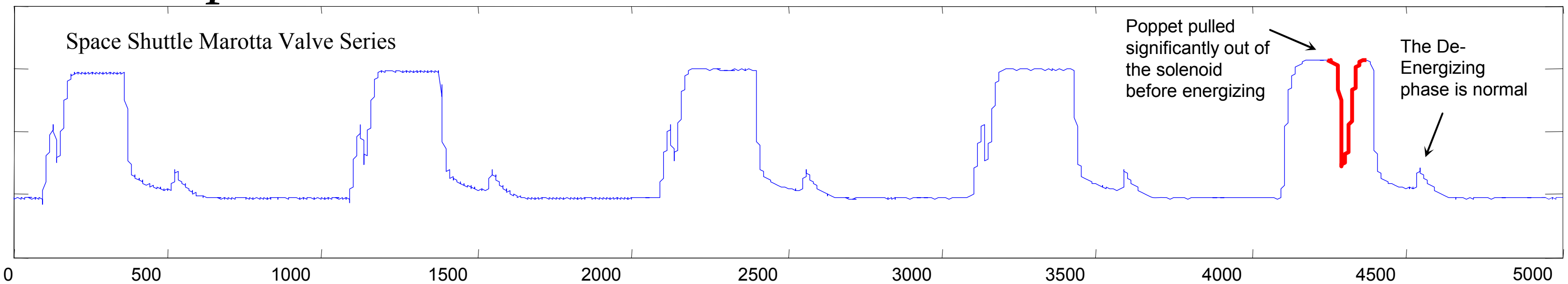


How was the discord able to find this very subtle Premature ventricular contraction? Note that in the normal heartbeats, the ST wave increases monotonically, it is only in the Premature ventricular contractions that there is an inflection.NB, this is not necessary true for all ECGS

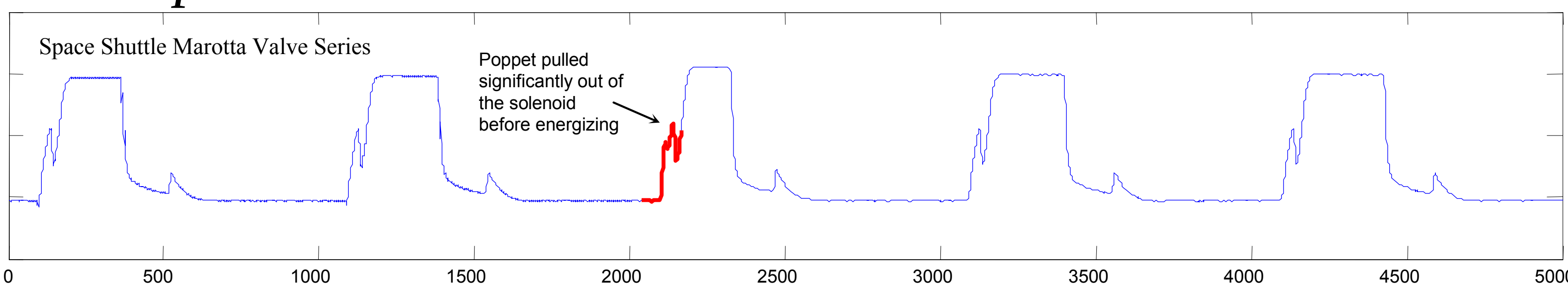


Discords in Space Shuttle Marotta Valve Series

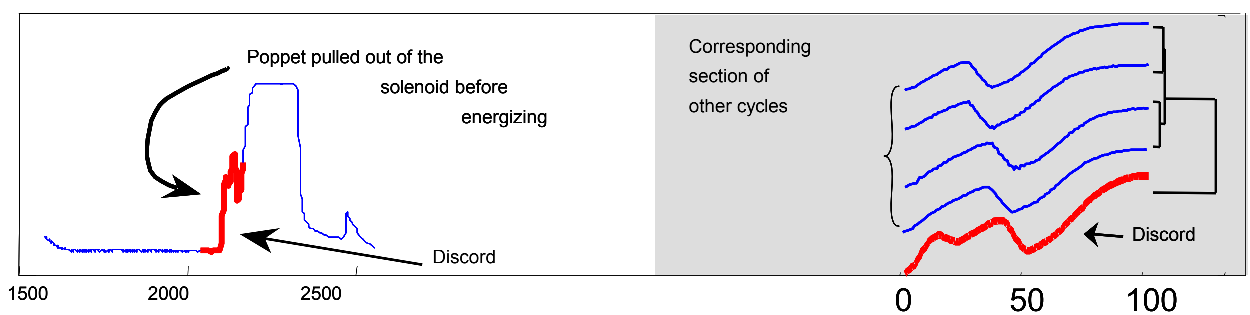
Example One



Example Two



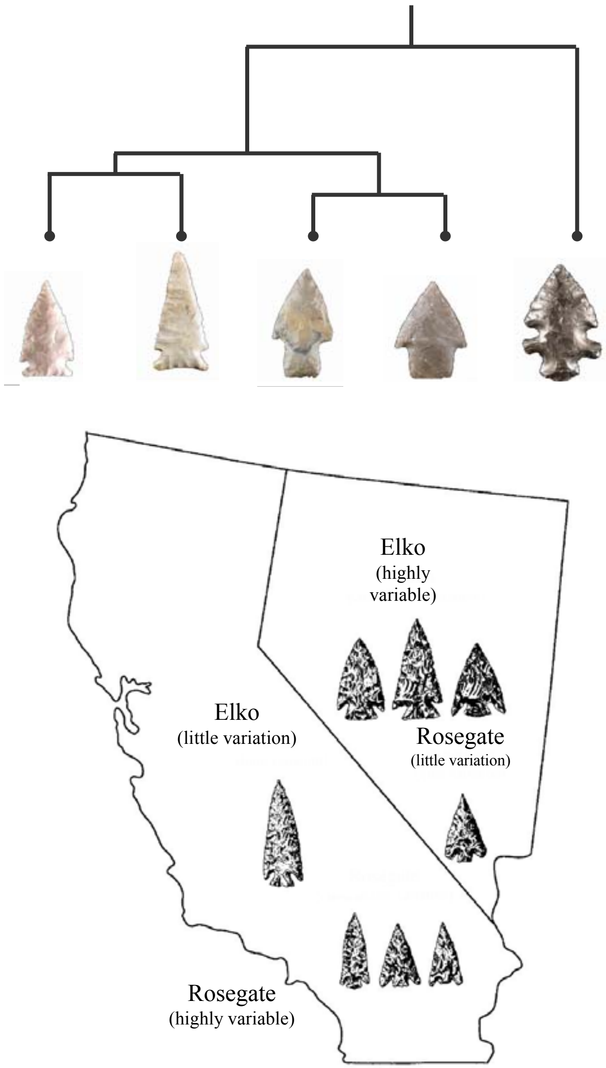
This discord is subtle, lets zoom in to see why it is a discord.

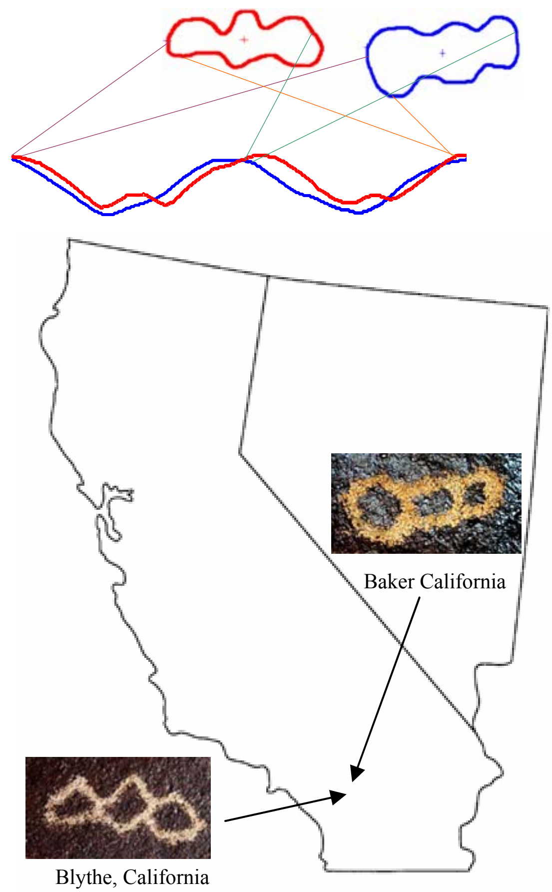


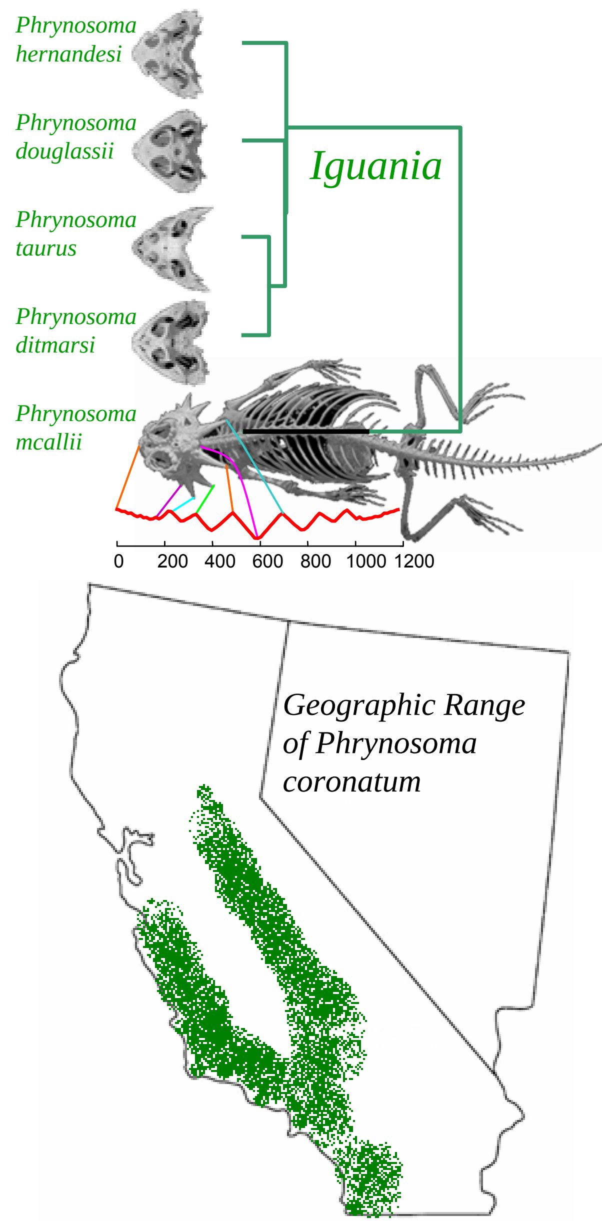
Open Problems

- Let us finish with a brief discussion of some open problems worthy of study

Spatially Constrained/Informed Mining of Shapes







Assessing the Significance of Motifs/Discords

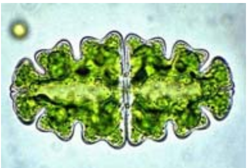
The motif and discord algorithms always return *some* answer, but is the result interesting, or something we should have expected by chance?

In a large string database, like this *ABBANBCJSMBAVSMABG*..
would it be more interesting to find…

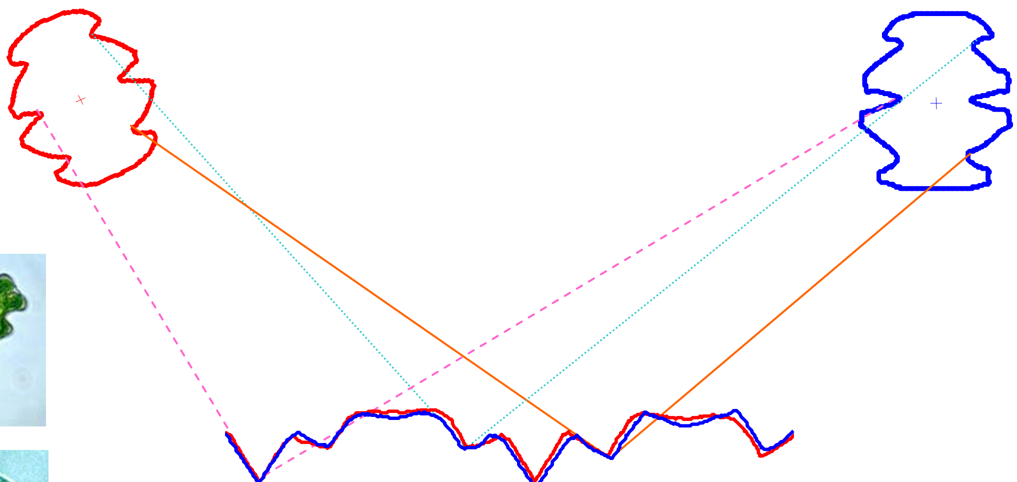
A motif pair {*ABBA*, *ABBA*}
A motif pair {*ABBAACCC*, *ABBBCCCC*}

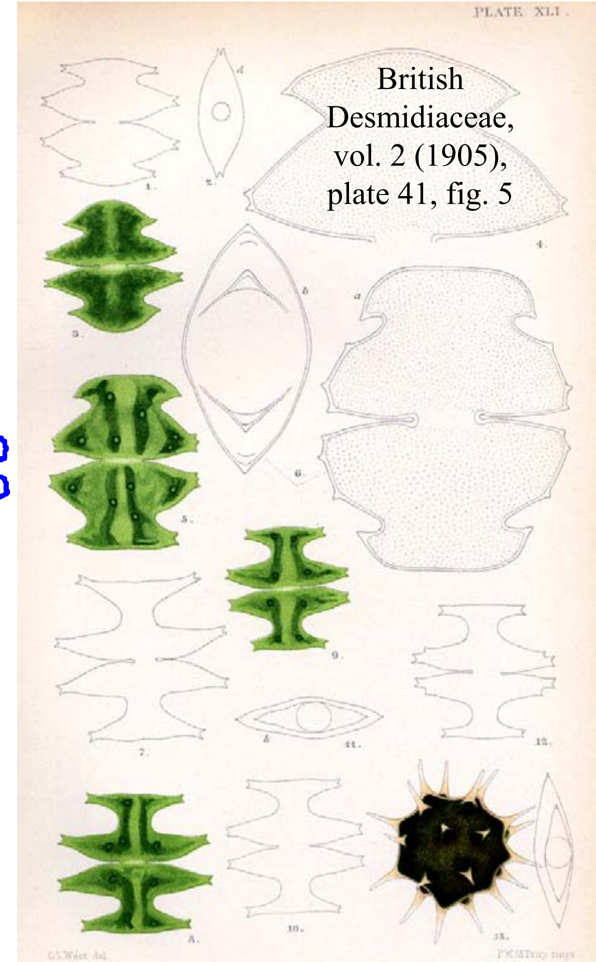
(i.e. shorter but perfect or longer with some misspellings)

Annotation of Historical Manuscripts

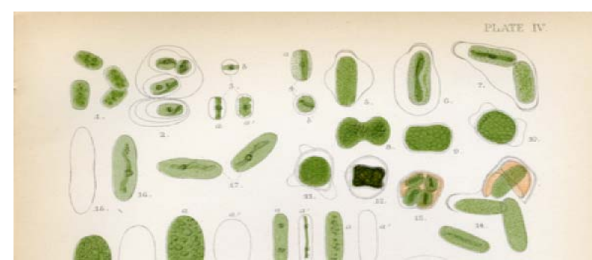


Micrasterias oscitans



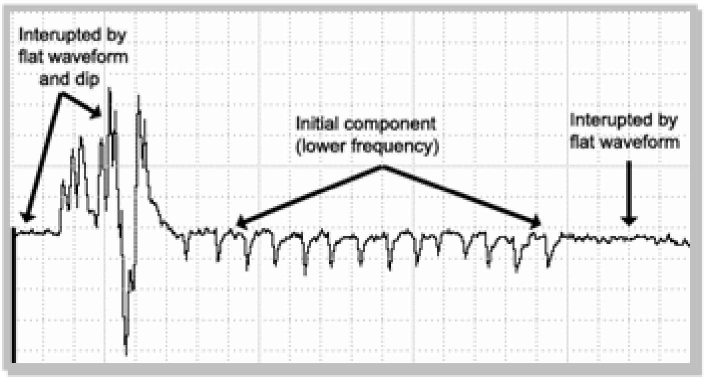


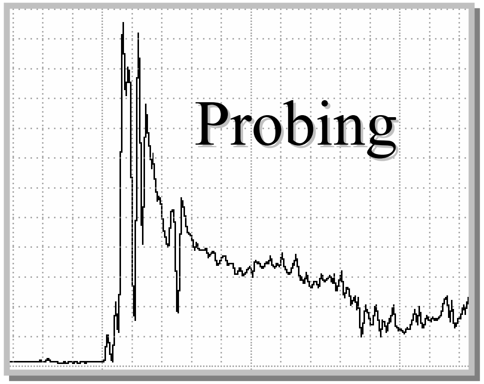
British Desmidiaceae, vol. 2 (1905), plate 41, fig. 5

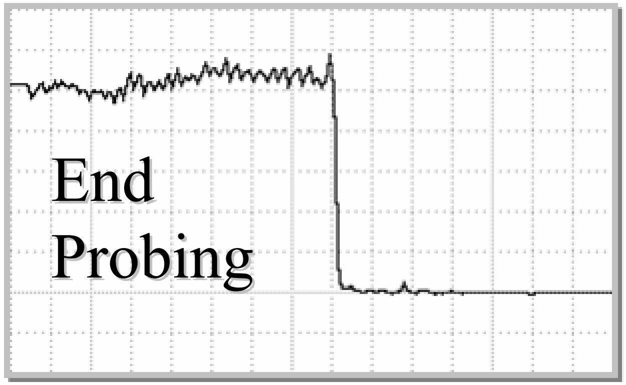


This match is based on shape only, the color and texture offer independent evidence

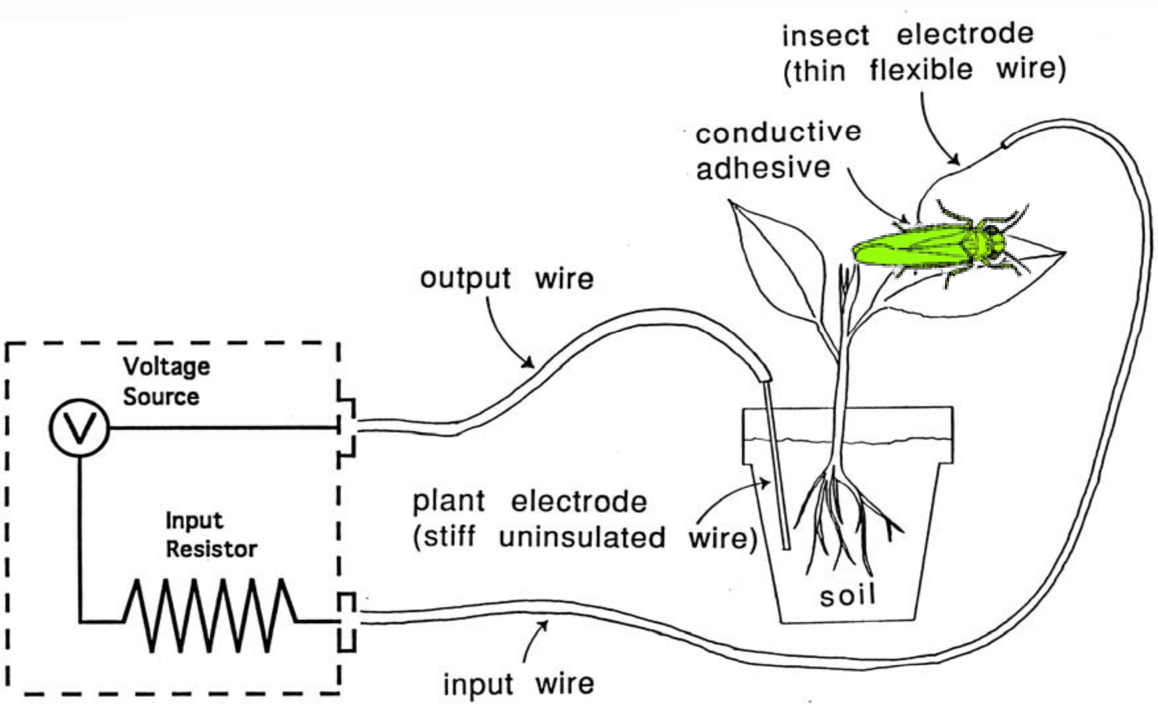
Applications!



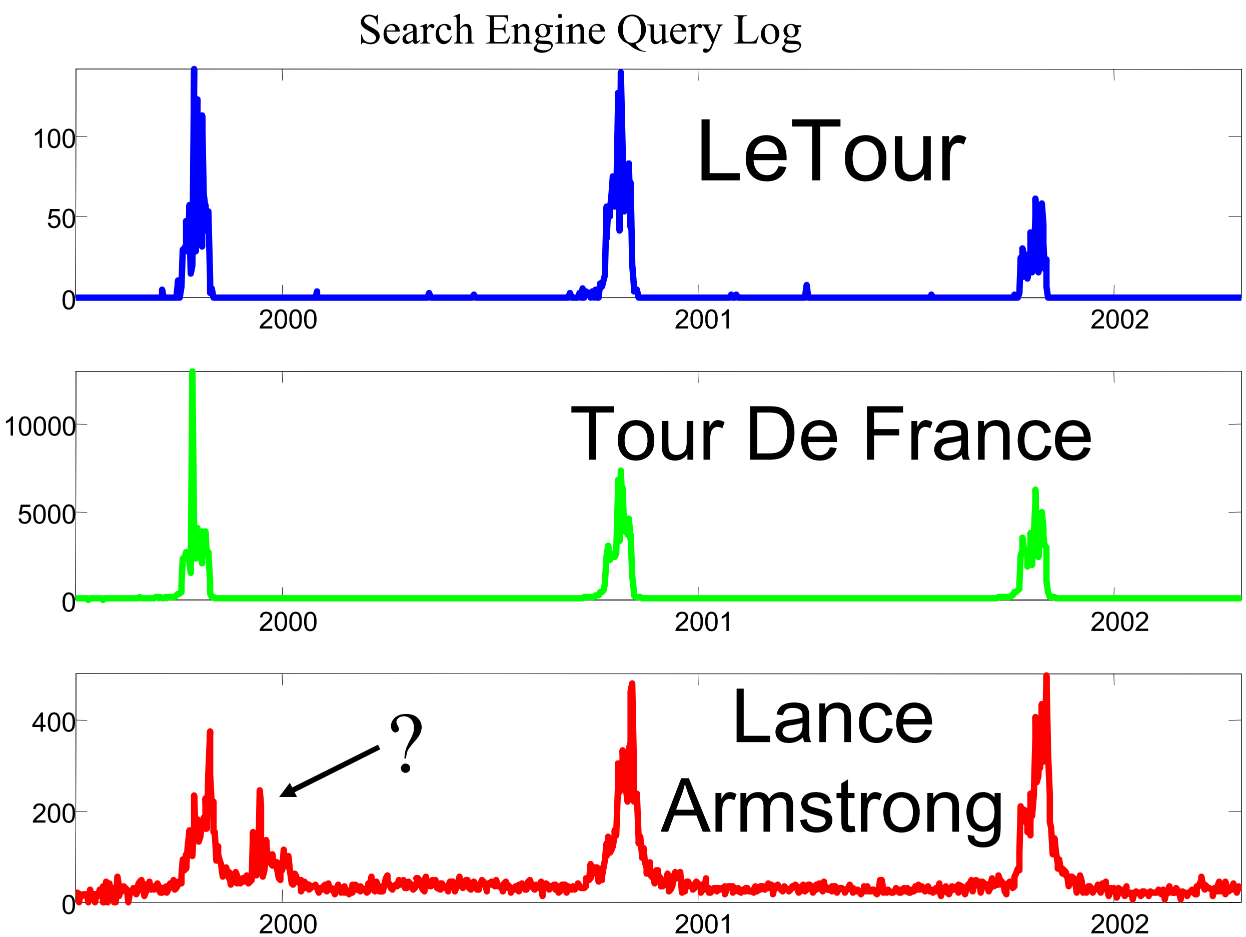




Beet Leafhopper, *Circulifer tenellus*



Mining Web Logs



It makes sense that the bursts for "LeTour", "Tour de France" and "Lance Armstrong" are all related.

But what caused the extra interest in Lance Armstrong in August/September 2000?

Example by M. Vlachos

The Last Word

The sun is setting on all other symbolic representations of time series, SAX is the *only* way to go


We are done!
We have seen that SAX is a very useful tool for solving problems in shape and time series data mining. I will be happy to answer any questions…
What are the disadvantages of using SAX
There are Nun

Thanks to my students

Appendix A

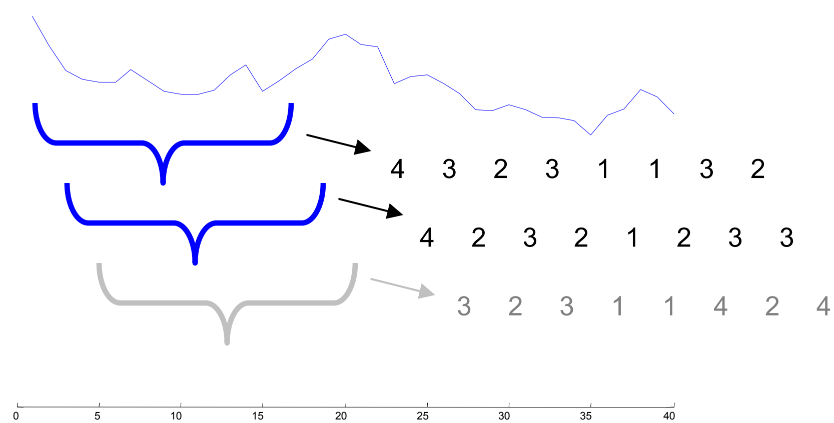
- Converting a long time series to a time series bitmap (Intelligent Icon)

```
>> x=random_walk(40,1);
>> timeseries2symbol(x, 16, 8, 4)

ans =

     4     3     2     3     1     1     3     2
     4     2     3     2     1     2     3     3
     3     2     3     1     1     4     2     4
     2     3     2     1     2     2     3     4
     2     2     1     1     3     2     3     4
     2     1     1     2     2     2     4     4
     2     1     1     3     1     3     4     4
     1     1     2     2     2     4     4     3
     1     1     3     1     3     4     4     2
     1     2     2     2     4     4     3     2
     1     2     1     3     4     4     2     2
     1     2     2     4     4     3     2     1
     3     1     3     4     4     2     2     1
     2     2     4     4     3     2     1     1
     3     3     4     4     2     2     1     1
     3     4     4     3     3     2     1     1
     3     4     4     2     2     1     1     1
     4     4     3     3     2     1     1     1
     4     4     3     3     2     2     1     1
     4     3     3     2     2     2     1     1
     4     4     3     2     2     1     1     2
     4     4     3     2     2     1     1     3
     4     4     2     2     1     1     2     3
     4     3     2     2     1     1     3     3
```

Just create random walk of length 40 for testing. Convert to SAX, with a sliding window of length 16, a word size of 8 and a cardinality of 4



```
>> x=random_walk(40,1);
>> timeseries2symbol(x, 16, 8, 4)

ans =

   G   T   C   T   A   A   T   C
   G   C   T   C   A   C   T   T
   T   C   T   A   A   G   C   G
   C   T   C   A   C   C   T   G
   C   C   A   A   T   C   T   G
   C   A   A   C   C   C   G   G
   C   A   A   T   A   T   G   G
   A   A   C   C   C   G   G   T
   A   A   T   A   T   G   G   C
   A   C   C   C   G   G   T   C
   A   C   A   T   G   G   C   C
   A   C   C   G   G   T   C   A
   T   A   T   G   G   C   C   A
   A   C   G   G   T   C   A   A
   C   T   G   G   C   C   A   A
   T   G   G   T   T   C   A   A
   T   G   G   C   C   A   A   A
   G   G   T   T   C   A   A   A
   G   G   T   T   C   C   A   A
   G   T   T   C   C   C   A   A
   G   G   T   C   C   A   A   C
   G   G   T   C   C   A   A   T
   G   G   C   C   A   A   C   T
   G   T   C   C   A   A   T   T
```

I have converted to “DNA” for visual clarity. Obviously we don’t really need to do this.

```
>> x=random_walk(40,1);
>> timeseries2symbol(x, 16, 8, 4)

ans =

    G   T   C   T   A   A   T   C
    G   C   T   C   A   C   T   T
    T   C   T   A   A   G   C   G
    C   T   C   A   C   C   T   G
    C   C   A   A   T   C   T   G
    C   A   A   C   C   C   G   G
    C   A   A   T   A   T   G   G
    A   A   C   C   C   G   G   T
    A   A   T   A   T   G   G   C
    A   C   C   C   G   G   T   C
    A   C   A   T   G   G   C   C
    A   C   C   G   G   T   C   A
    T   A   T   G   G   C   C   A
    A   C   G   G   T   C   A   A
    C   T   G   G   C   C   A   A
    T   G   G   T   T   C   A   A
    T   G   G   C   C   A   A   A
    G   G   T   T   C   A   A   A
    G   G   T   T   C   C   A   A
    G   T   T   C   C   C   A   A
    G   G   T   C   C   A   A   C
    G   G   T   C   C   A   A   T
    G   G   C   C   A   A   C   T
    G   T   C   C   A   A   T   T
```

Count the frequency of all pair of basepairs.

Below I have just done **AA** and **AC**

Assign the results to a matrix z

AA	AC	CA	CC
AG	AT	CG	CT
GA	GC	TA	TC
GG	GT	TG	TT

z =

19	10	CA	CC
AG	AT	CG	CT
GA	GC	TA	TC
GG	GT	TG	TT

We need to normalize the matrix z, below is *one* way to do it such that the min value is 0 and the max values is 1. (matlab code)

There may be better ways to normalize…

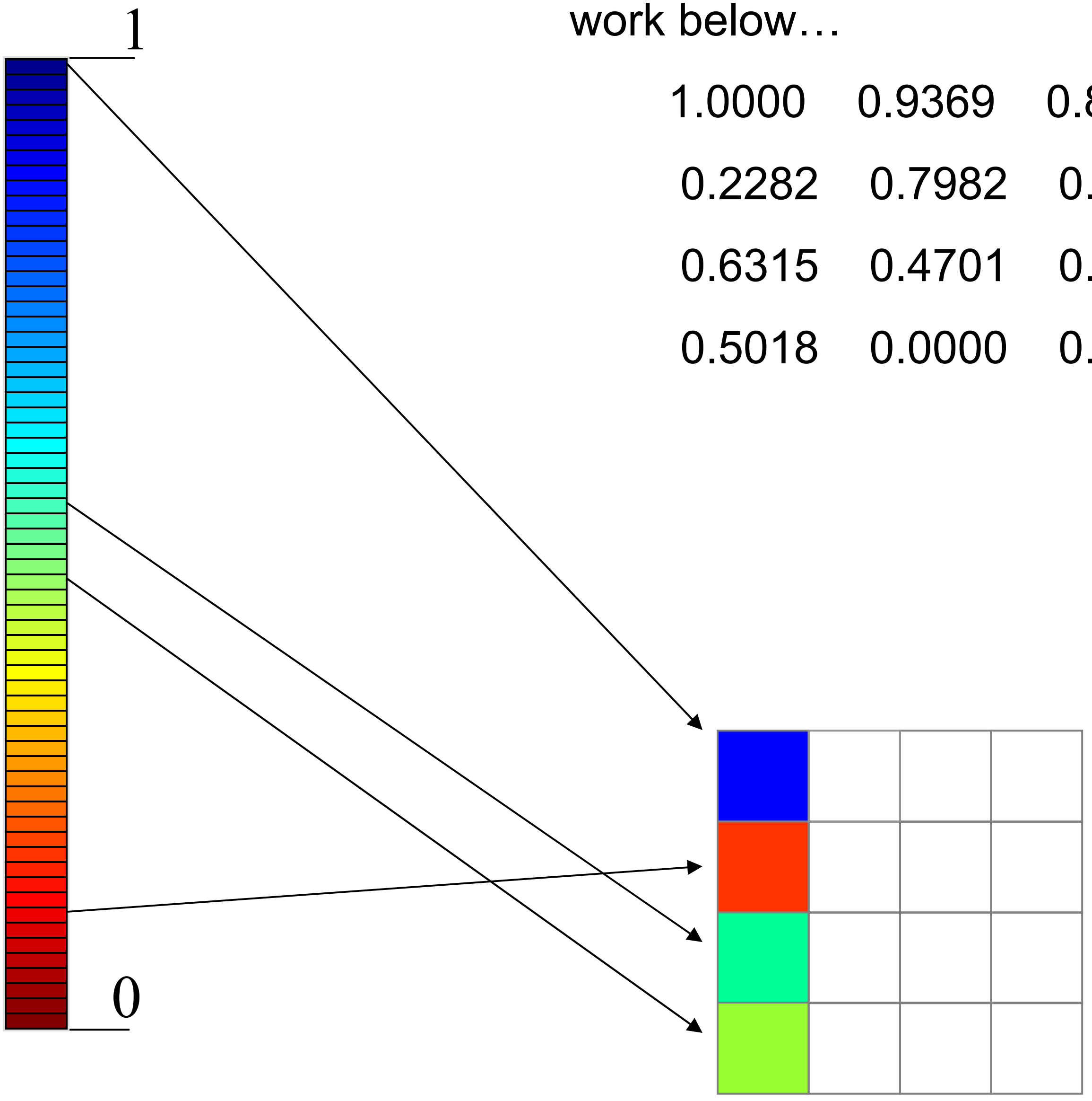
```
>> z=(z-min(min(z)));
>> z=(z/max(max(z)))
```

z =

1.0000	0.9369	0.8618	0.9696
0.2282	0.7982	0.4575	0.7725
0.6315	0.4701	0.6407	0.1693
0.5018	0.0000	0.8302	0.4156

Map to some colormap, I have done ¼ of the work below…

1.0000	0.9369	0.8618	0.9696
0.2282	0.7982	0.4575	0.7725
0.6315	0.4701	0.6407	0.1693
0.5018	0.0000	0.8302	0.4156



Hints I

ans =

G	T	C	T	**A**	**A**	T	C
G	C	T	C	A	C	T	T
T	C	T	A	A	G	C	**A**
A	T	C	A	C	C	T	G
C	C	**A**	**A**	T	C	T	G

When counting patterns, don't count patterns that span two lines.

For example, don't count the underlined A's as an occurrence of **AA**

Hints II

ans =

G	T	C	T	A	A	T	C
G	T	C	T	A	A	T	C
G	C	T	C	A	C	T	T
T	C	T	A	A	G	C	A
A	T	C	A	C	C	T	G
C	C	A	A	T	C	T	G

Note that here lines 1 and 2 are the same. This can happen a lot, especially with smooth time series and/or a high compression ratio.

The SAX code has an extra parameter that removes these redundant lines. It seems like this makes the Intelligent Icons work better, and it does make the code run a little faster.

Hints III

For Intelligent Icon the cardinality must be 4

But what is the best sliding window length?

What is the best a word size?

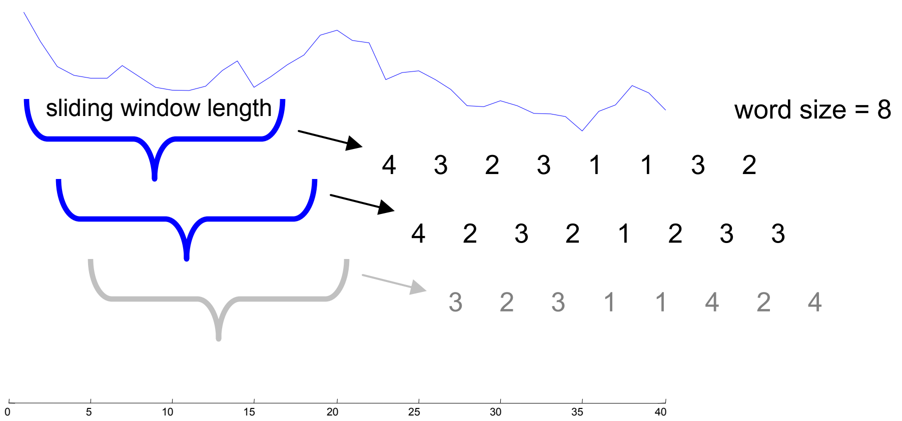
At the moment there is no answer to this other than playing with the data (or CV if you have labeled data)

The good news is that once you find good settings for your domain (say ECGs) then the settings should work for all ECGS.

Heuristics:

The sliding window length should be about twice the length of the natural scale at which the data is interesting. For example, about two heartbeats for cardiology, or for power demand, about two days.

The smoother the data, the smaller you can make the word size.



Appendix: DTW

- There are some critical facts about the size of the warping window r.
- r can vary from 0% (the special case of Euclidian distance) to 100% (the special case of full DTW).
- Without lower bounding, the time taken is approximately linear in r, so r =5% is about twice as fast as r =10%.
- With lower bounding, the time taken is highly non-linear in r, so r =5% is perhaps 10 to 100 times as fast as r =10%.
- In general (empirically measured over 35 datasets) the following is true.
- If you start with r = 0 and you make it larger, the accuracy improves, then gets worse (see the two examples for FACE and GUN in this tutorial, but it is true for other datasets)
- The best accuracy tends to be at a relatively small value for r (usually just 2 to 5%)
- For any dataset, the best value for r depends on the size of the training set. For example for CBF with just 20 instances, you might need r = 8%, but with 200 instances you only need 1 or 2%, and with 2,000 instances, you need r = 0% (the Euclidean distance).
- How do you find the best choice for r? Use cross valuation to test for the best value.

See [a] and [b]

[a] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei & Chotirat Ann Ratanamahatana (2006). Fast Time Series Classification Using Numerosity Reduction. ICML

[b] Ratanamahatana, C. A. and Keogh. E. (2004). Everything you know about Dynamic Time Warping is Wrong. Third Workshop on Mining Temporal and Sequential Data, in conjunction with the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), August 22-25, 2004 - Seattle, WA