Mining Shape and Time Series Databases

CIS 8590 Perception of Intelligent Systems

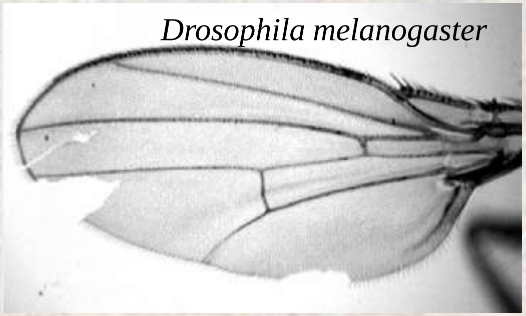
Slides created by:
Eamonn Keogh eamonn@cs.ucr.edu
Suzan Köknar-Tezel

Outline of Tutorial I

- Introduction, Motivation
- The ubiquity of time series and shape data
- What are time series?
- Examples of problems in time series and shape data mining
- How to define “similar”
- Shape Representation
- Properties of distance measures
  - Euclidean distance
  - Dynamic time warping
  - Longest common subsequence
- Searching quickly
- Spatial Access Methods and the curse of dimensionality
- Generic dimensionality reduction
- Some real-world problems
- Our work - OSB

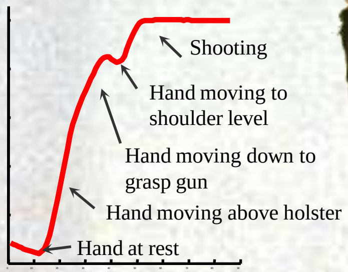
The Ubiquity of Shape

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

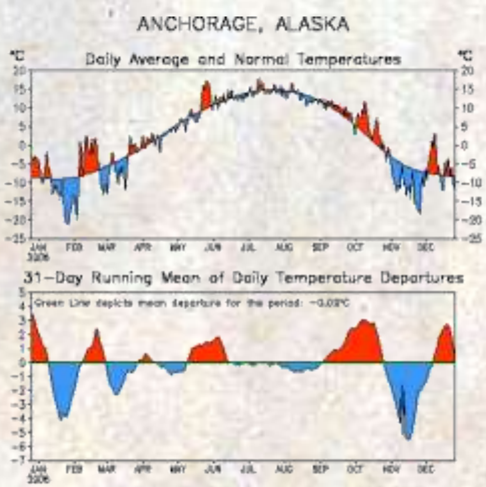


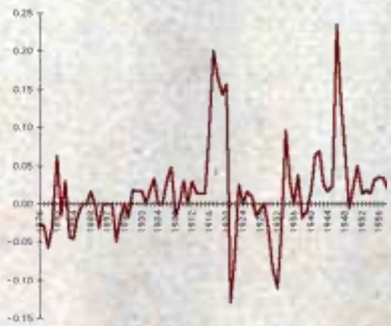
Drosophila melanogaster

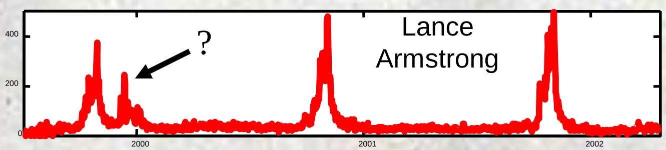
The Ubiquity of Time Series

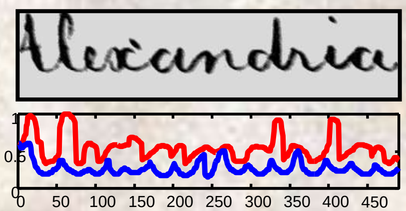


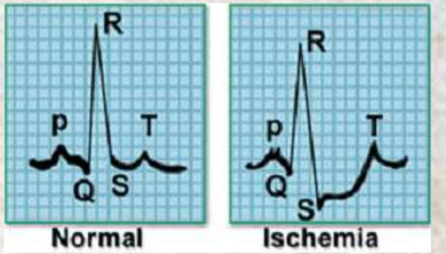
Don't Shoot! *Motion capture, meteorology, finance, handwriting, medicine, web logs, music…*











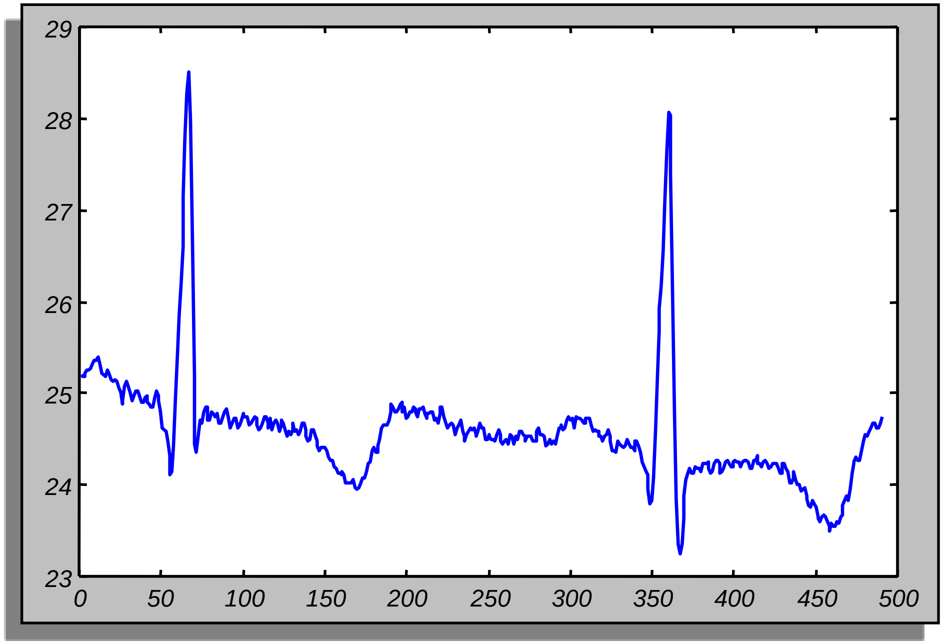
25.1750
25.2250
25.2500
25.2500
25.2750
25.3250
25.3500
25.3500
25.4000
25.4000
25.3250
25.2250
25.2000
25.1750

..

..
24.6250
24.6750
24.6750
24.6250
24.6250
24.6250
24.6750
24.7500

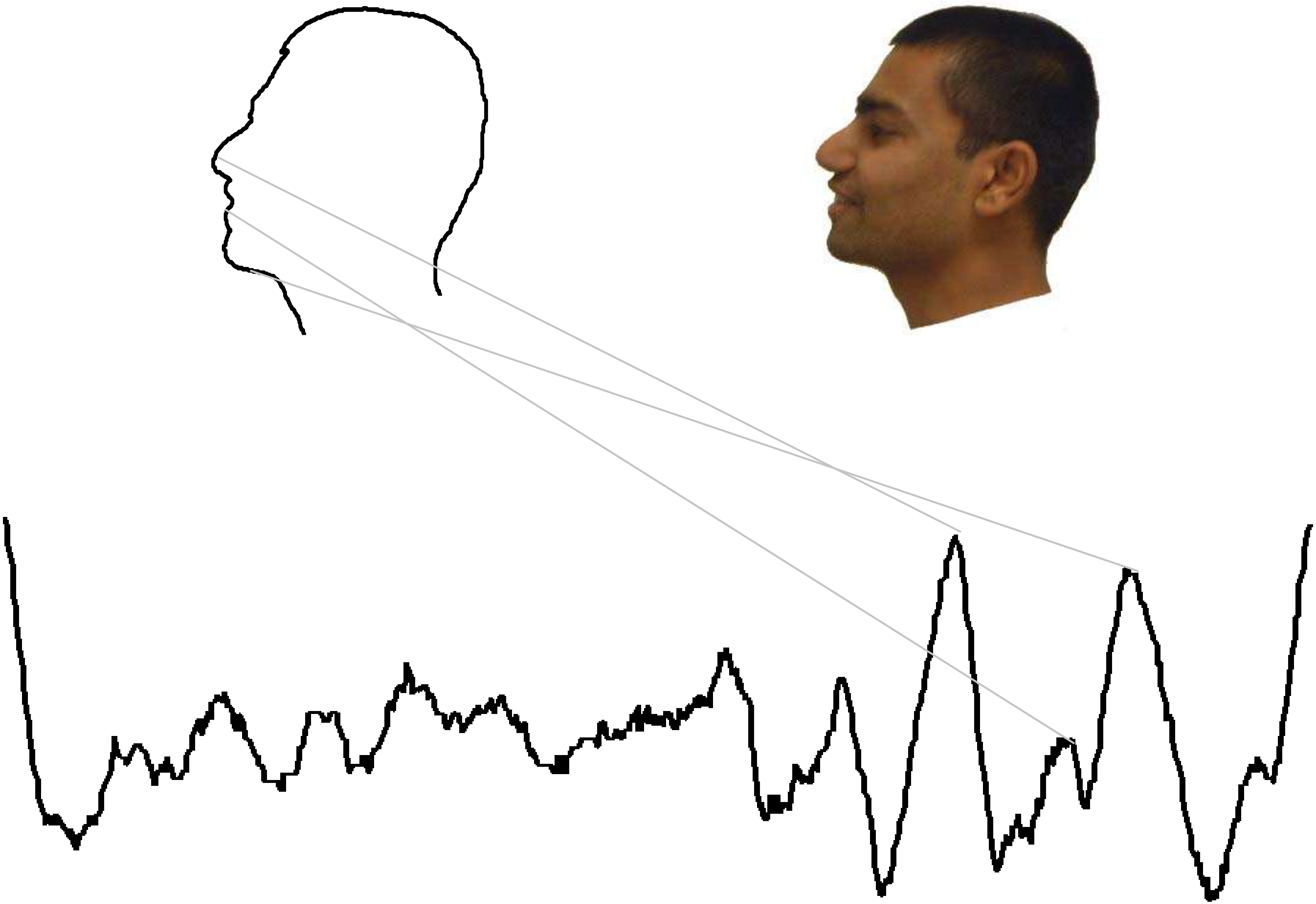
What are Time Series?

A time series is a collection of observations made sequentially in time.



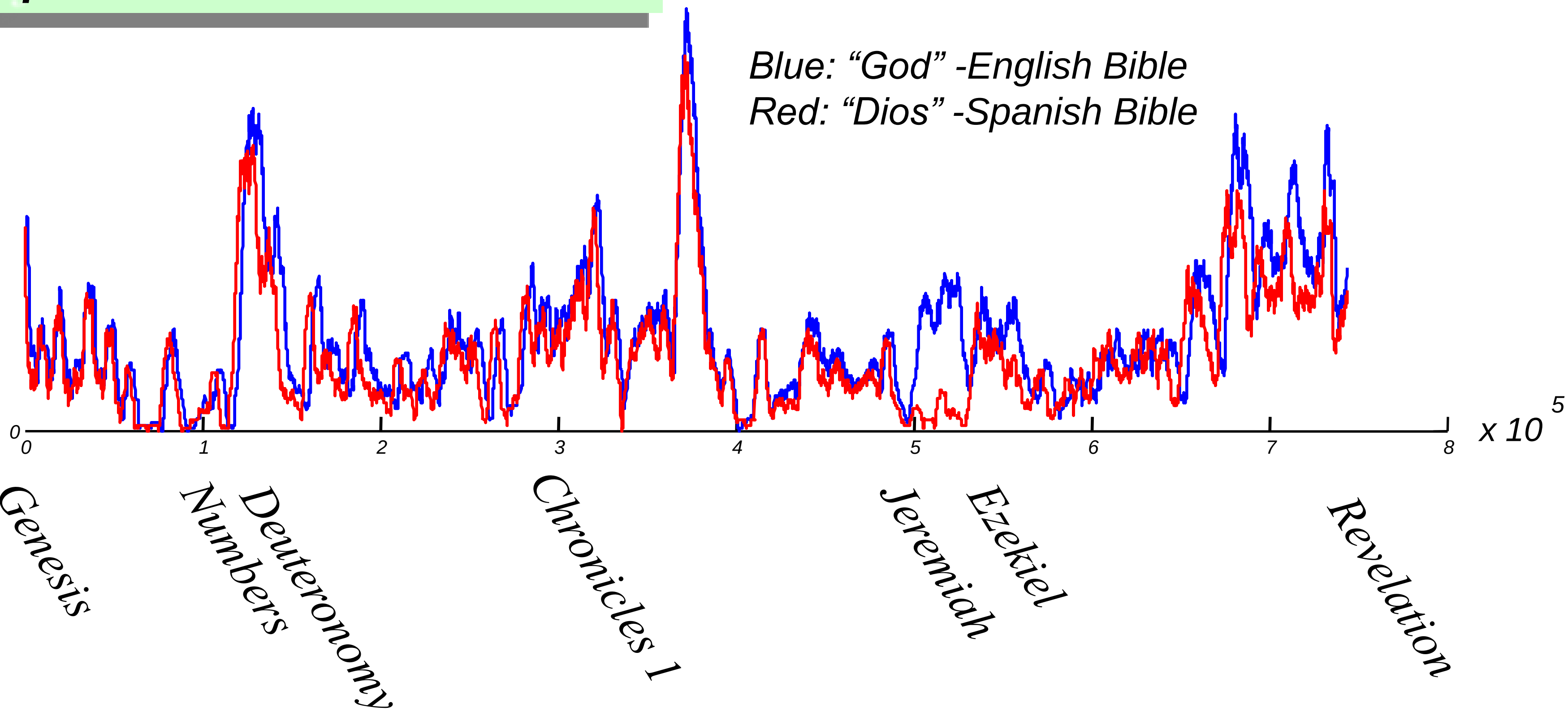
Virtually all similarity measurements and dimensionality reduction techniques discussed in this tutorial can be used with other data types

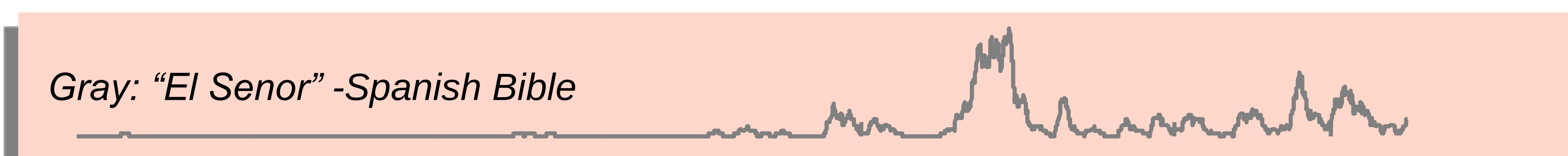
Image data, may best be thought of as time series…



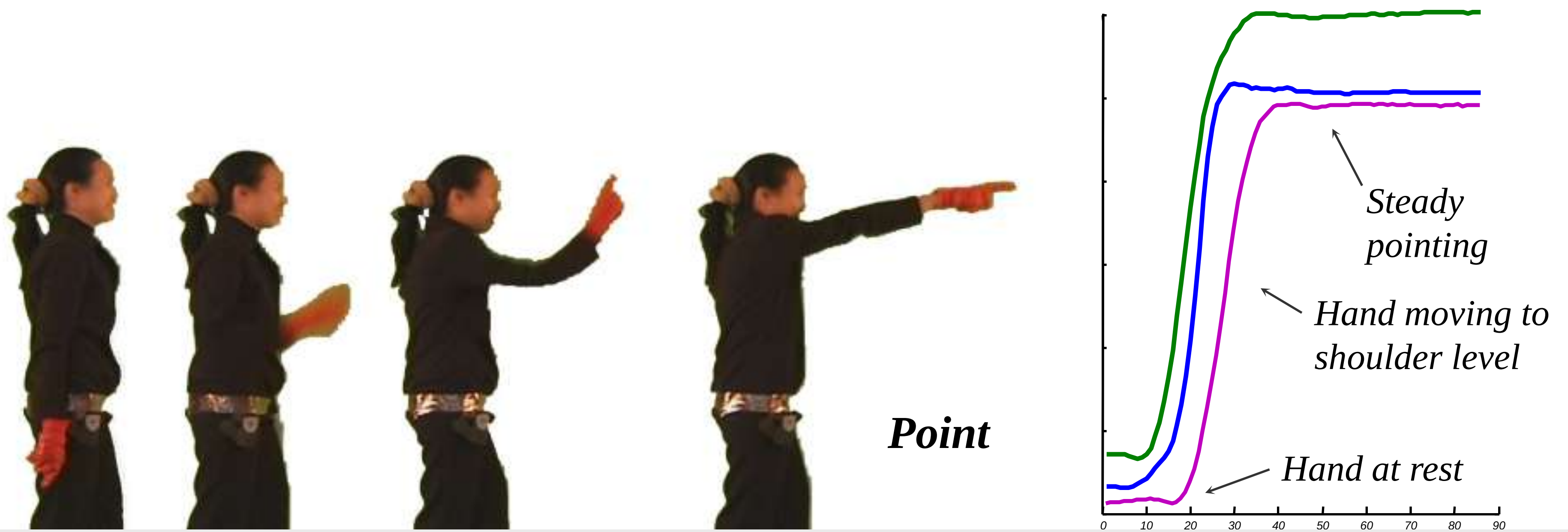
Text data, may best be thought of as time series…

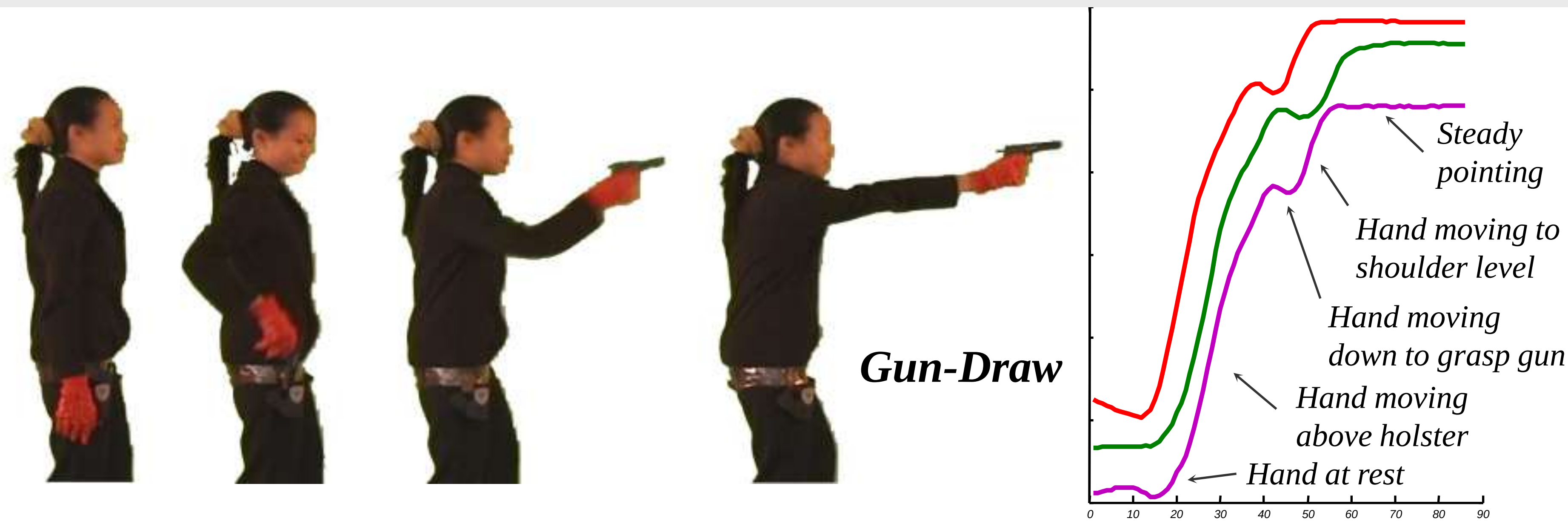
The local frequency of words in the Bible





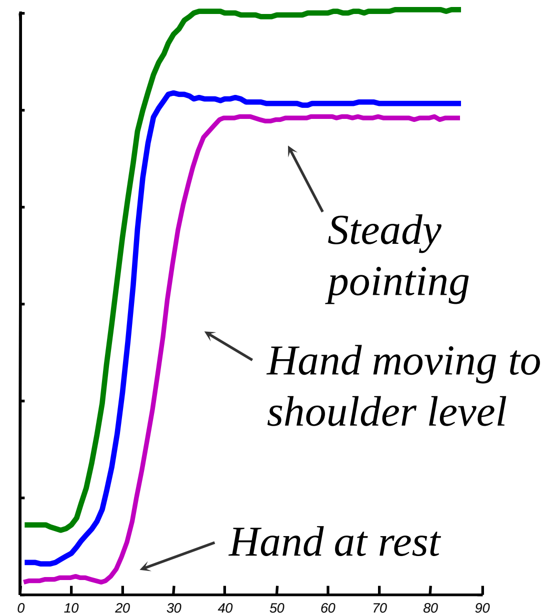
Video data, may best be thought of as time series…



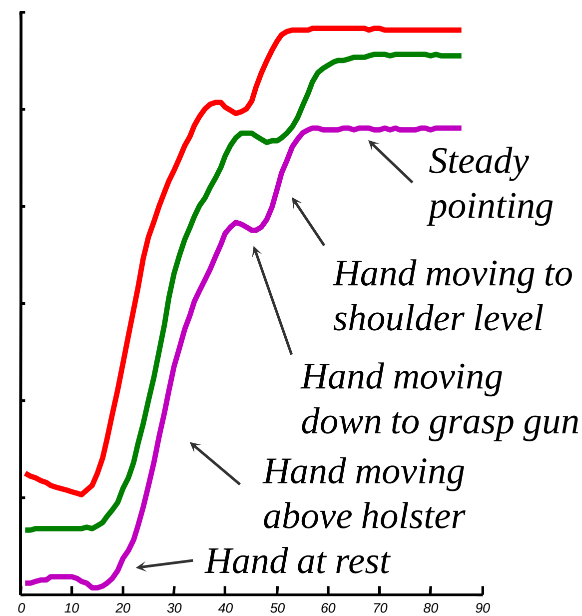


Video data, may best be thought of as time series…

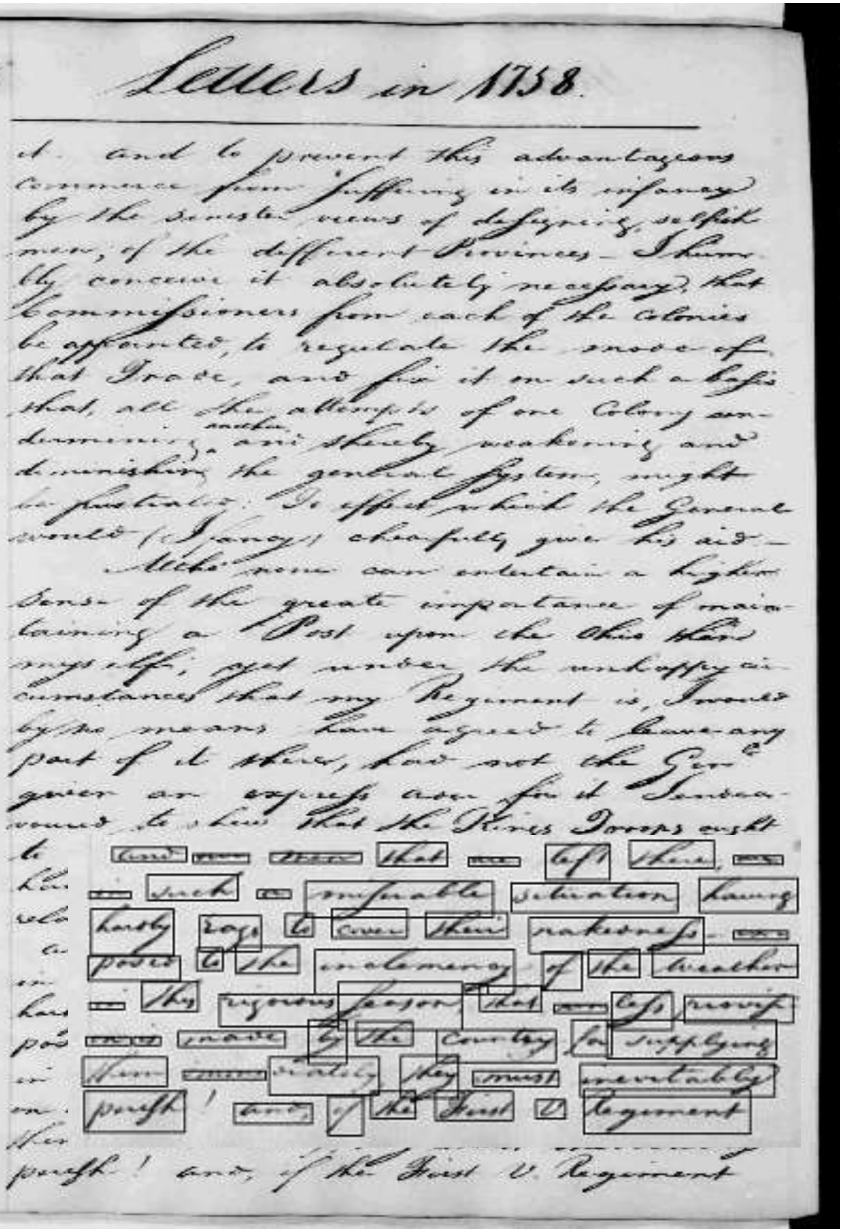
Point



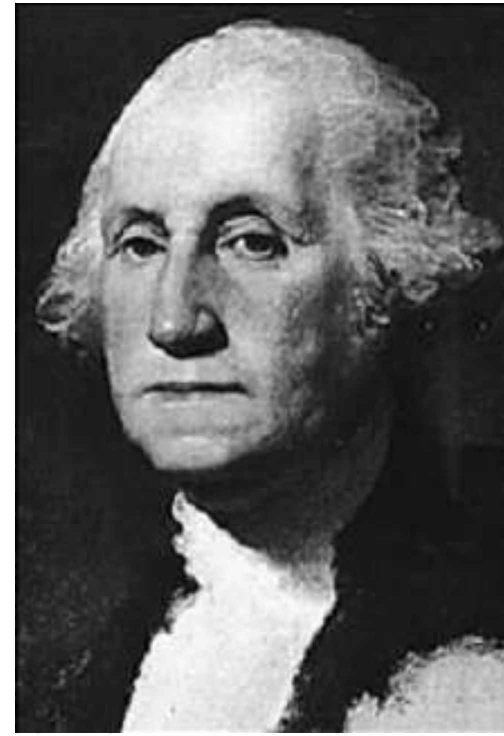
Gun



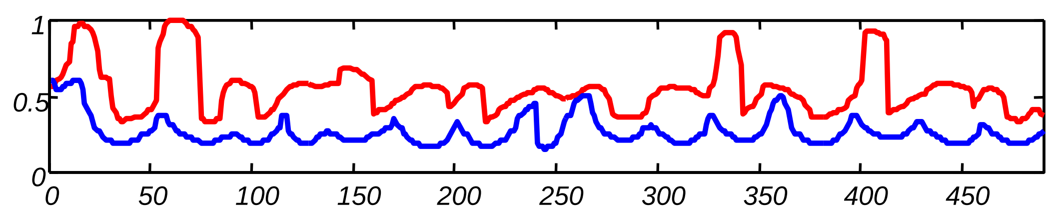
Handwriting data, may best be thought of as time series…



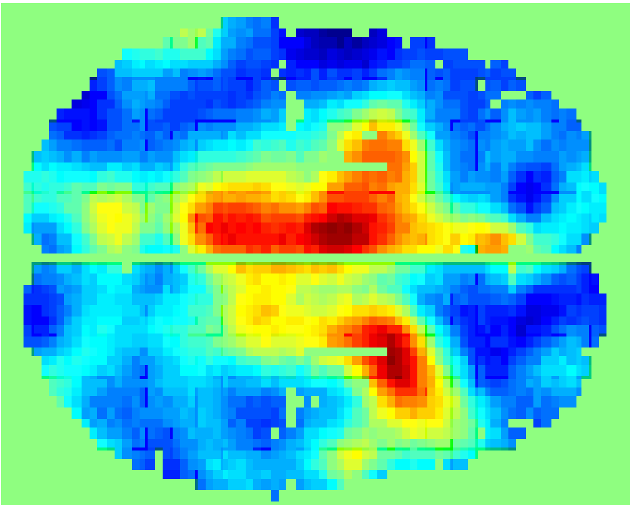
George Washington Manuscript

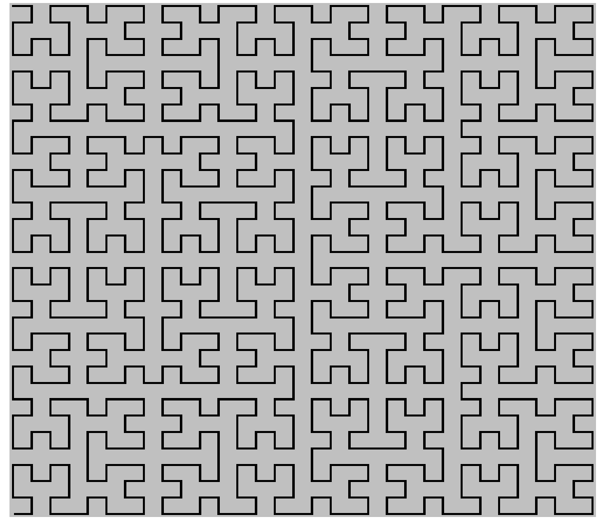


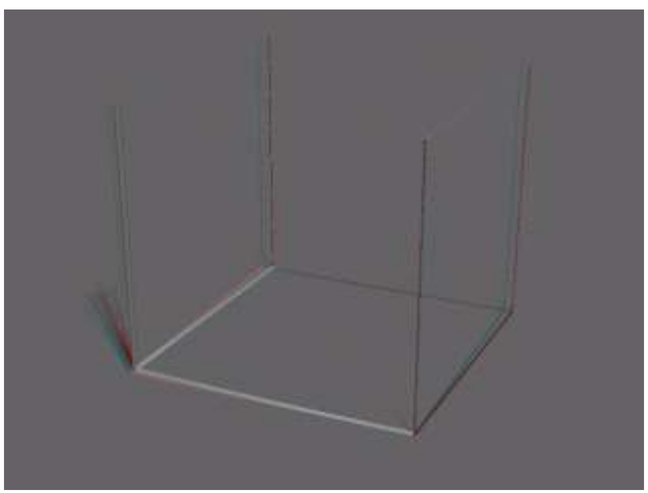
George Washington
1732-1799

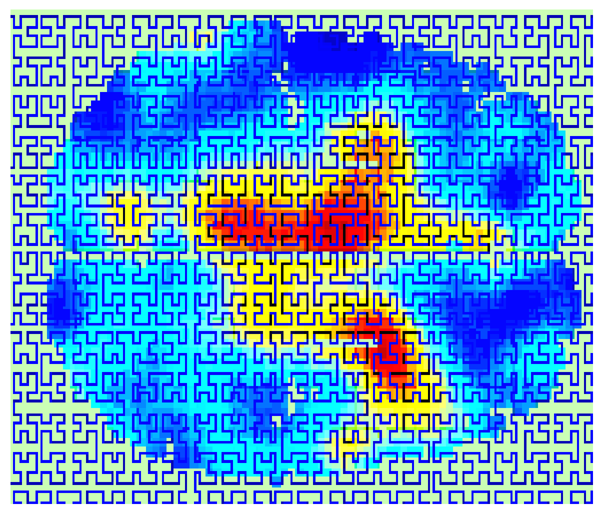


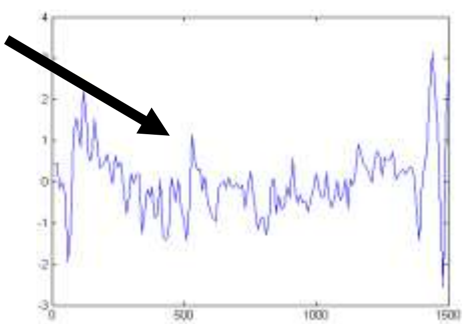
Brain scans (3D voxels), may best be thought of as time series…







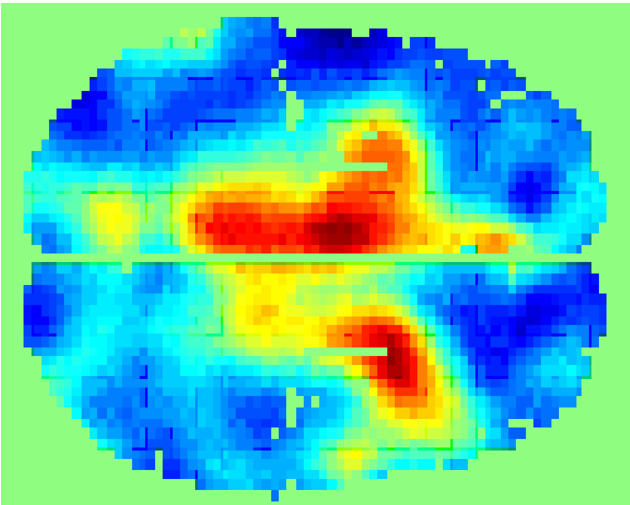


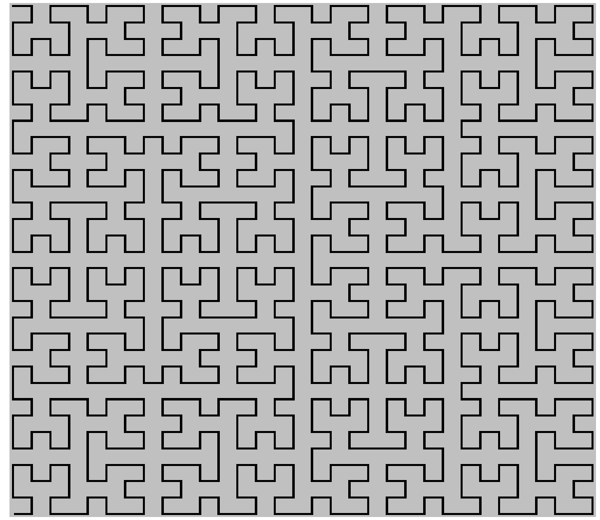


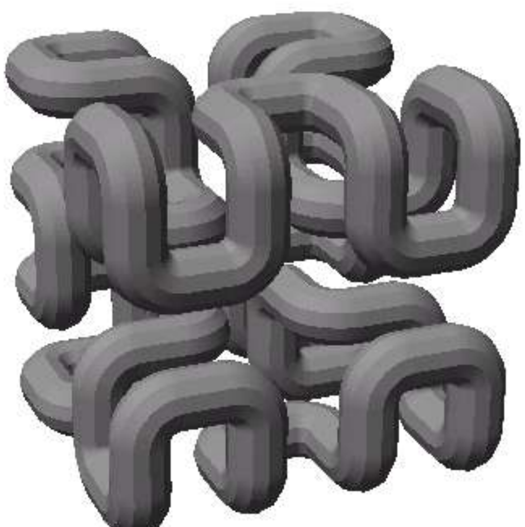
Works with
3D glasses!

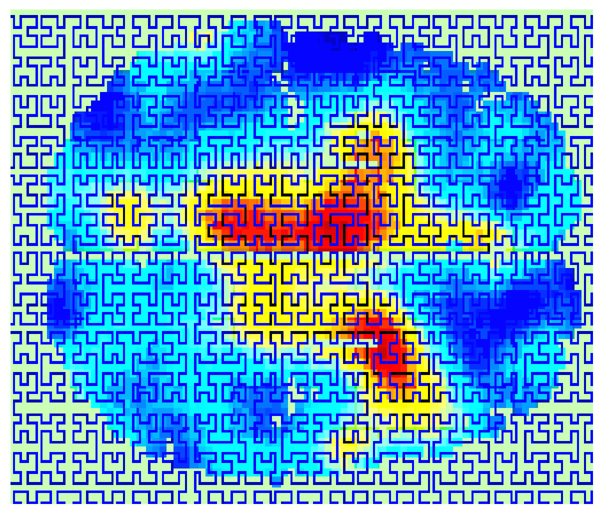
Wang, Kontos, Li and Megalooikonomou ICASSP 2004

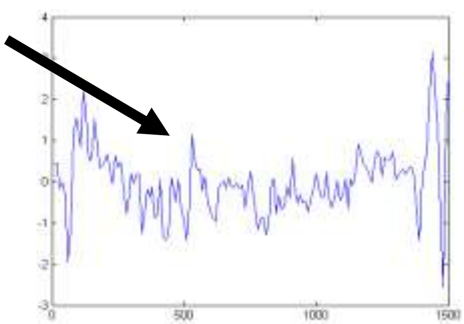
Brain scans (3D voxels), may best be thought of as time series…











Wang, Kontos, Li and Megalooikonomou ICASSP 2004

Why is Working With Time Series so Difficult? Part I

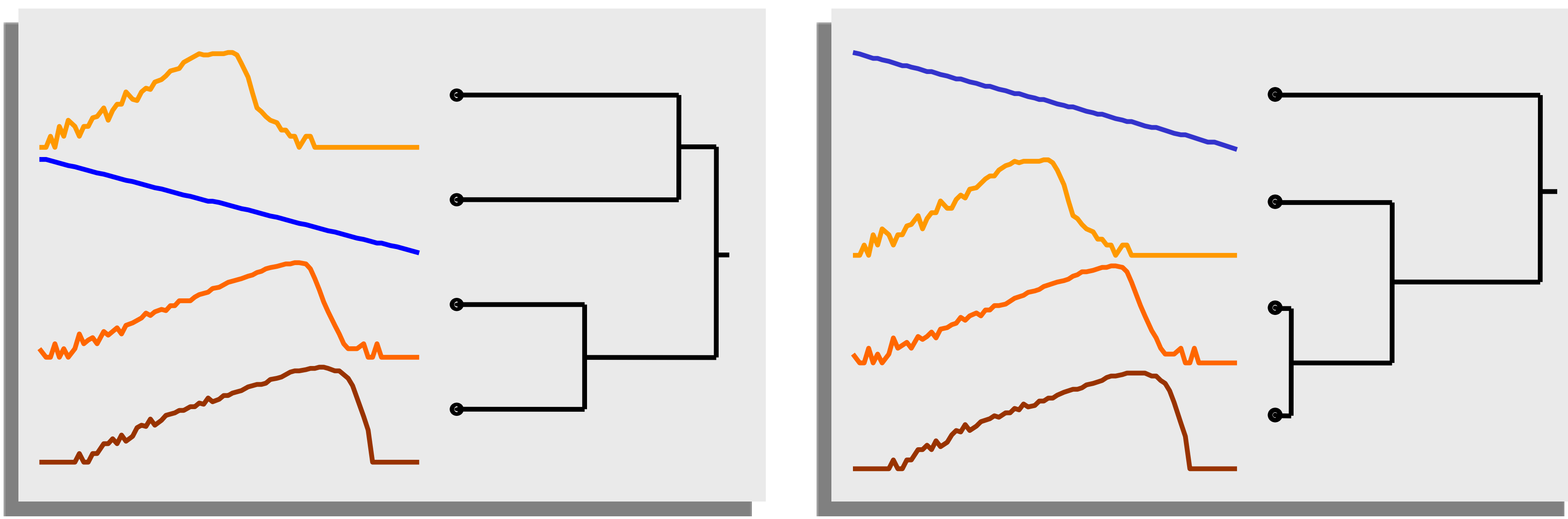
Answer: *How do we work with very large databases?*

- *1 Hour of EKG data: 1 Gigabyte.*
- *Typical Weblog: 5 Gigabytes per week.*
- *Space Shuttle Database: 200 Gigabytes and growing.*
- *Macho Database: 3 Terabytes, updated with several gigabytes per night.*

Since most of the data lives on disk (or tape), we need a representation of the data we can efficiently manipulate.

Why is Working With Time Series so Difficult? Part II

Answer: *We are dealing with subjectivity*



The definition of similarity depends on the user, the domain and the task at hand. We need to be able to handle this subjectivity.

Why is working with time series so difficult? Part III

Answer: *Miscellaneous data handling problems.*

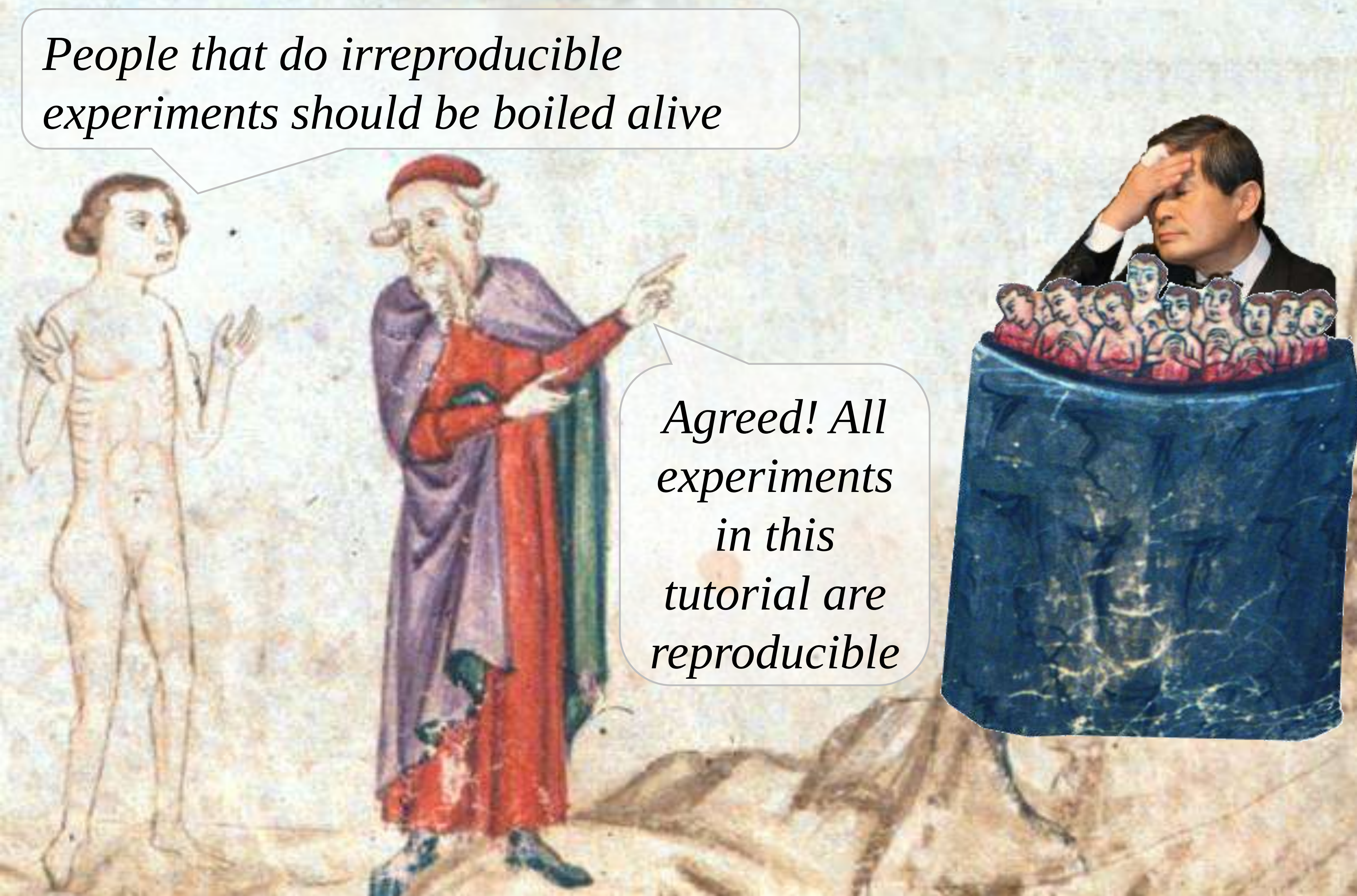
- *Differing data formats.*
- *Differing sampling rates.*
- *Noise, missing values, etc.*

We will not focus on these issues in this tutorial.

Examples of problems in time series and shape data mining

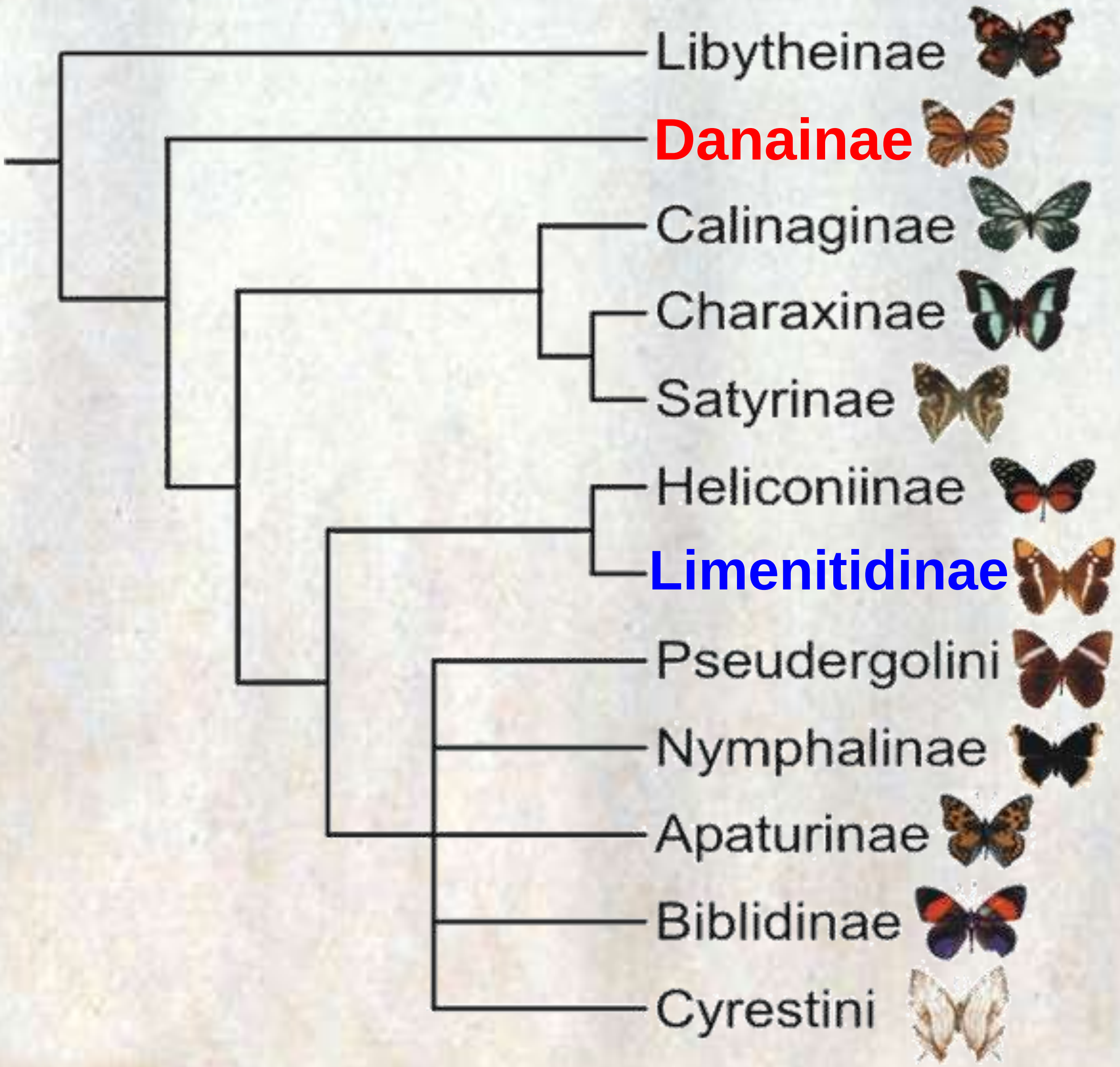
In the next few slides we will see examples of the kind of problems we would like to be able to solve, then later we will see the necessary tools to solve them

All our Experiments are Reproducible!

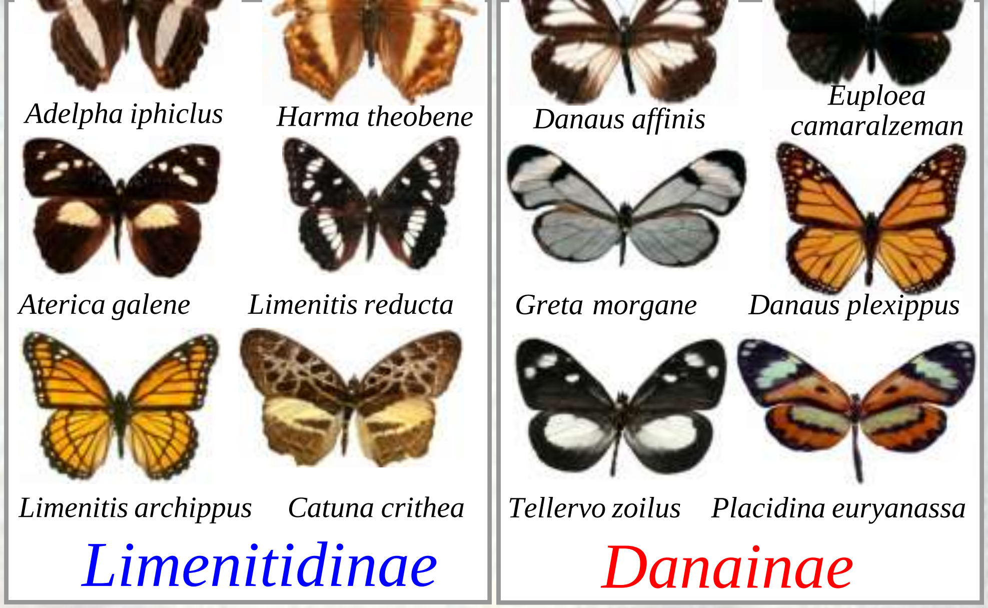


Example 1: Join

Given two data collections, link items occurring in each

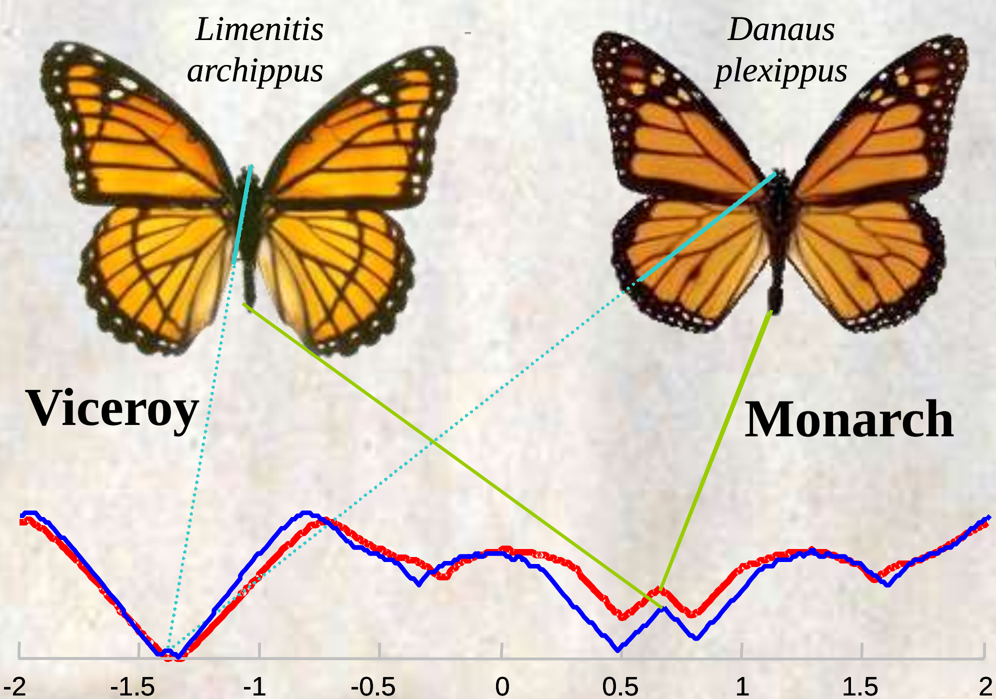


We can take two different families of butterflies, Limenitidinae and Danainae, and find the most similar shape between them



*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**

Example 2: Annotation

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

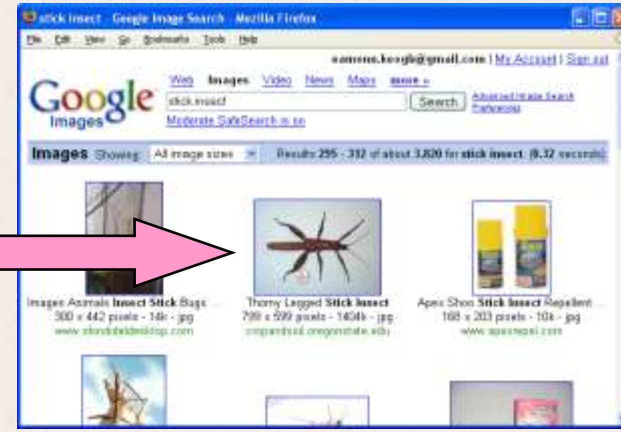
Friedrich Bertuch's *Bilderbuch fur Kinder* (Weimar, 1798–1830)

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?

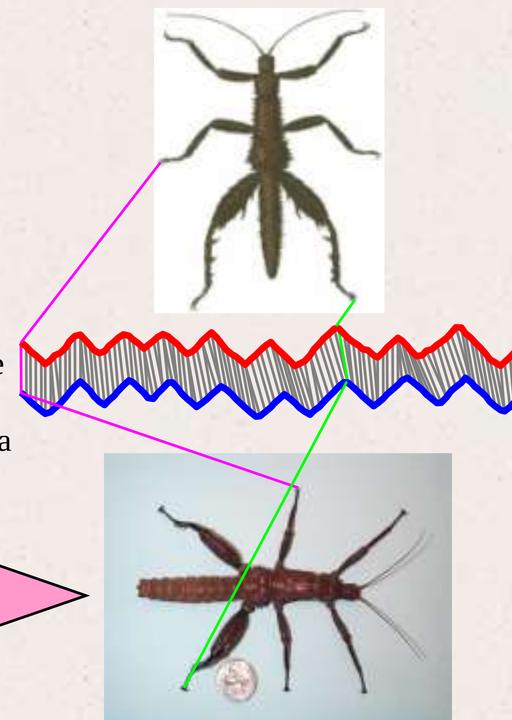
Suppose we issue a query to Google search for "*Stick Insect*" and further filter the results by shape similarity….



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

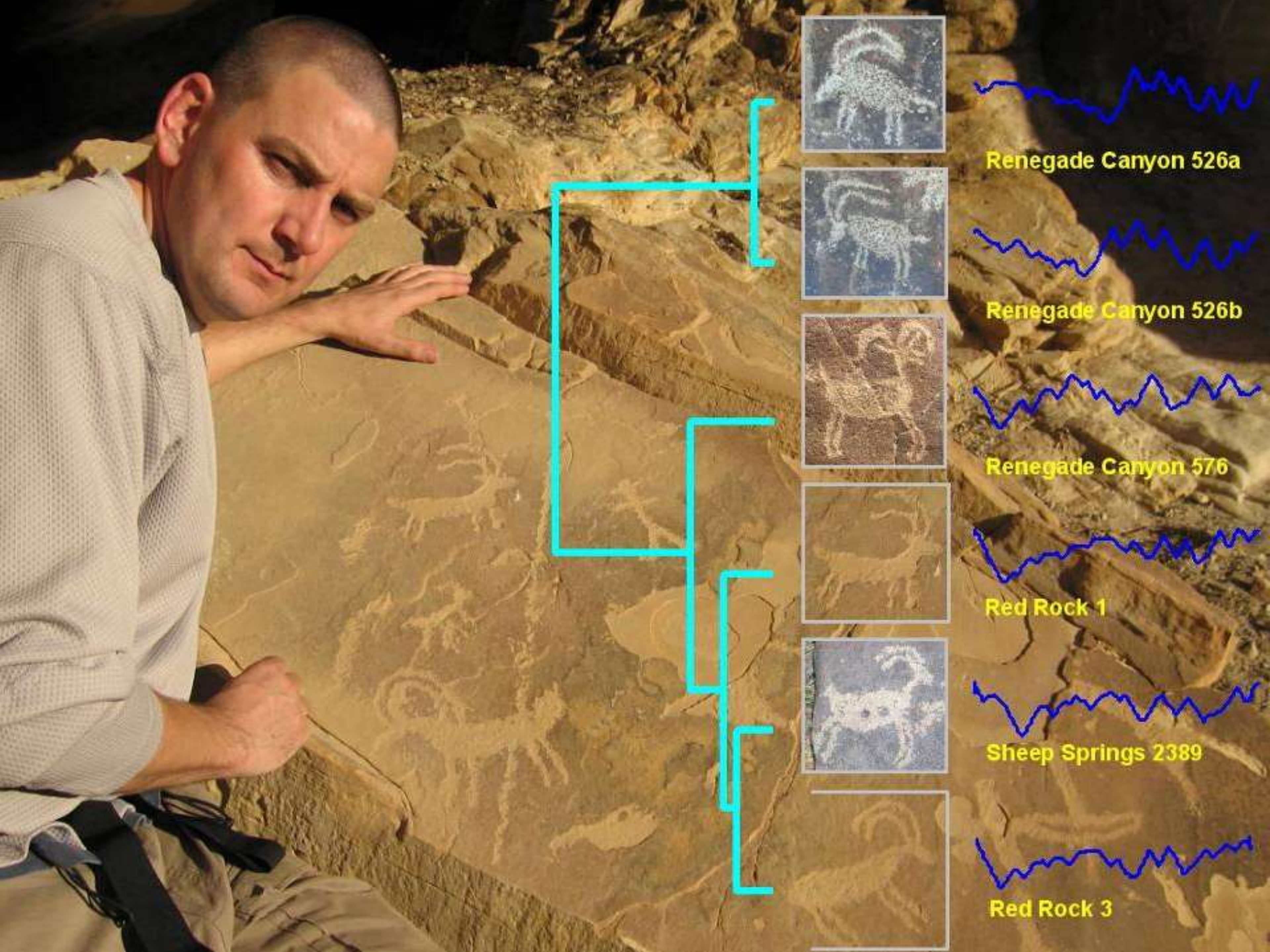
- They appear worldwide
- Over a million in America alone
- Surprisingly little known about them

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**

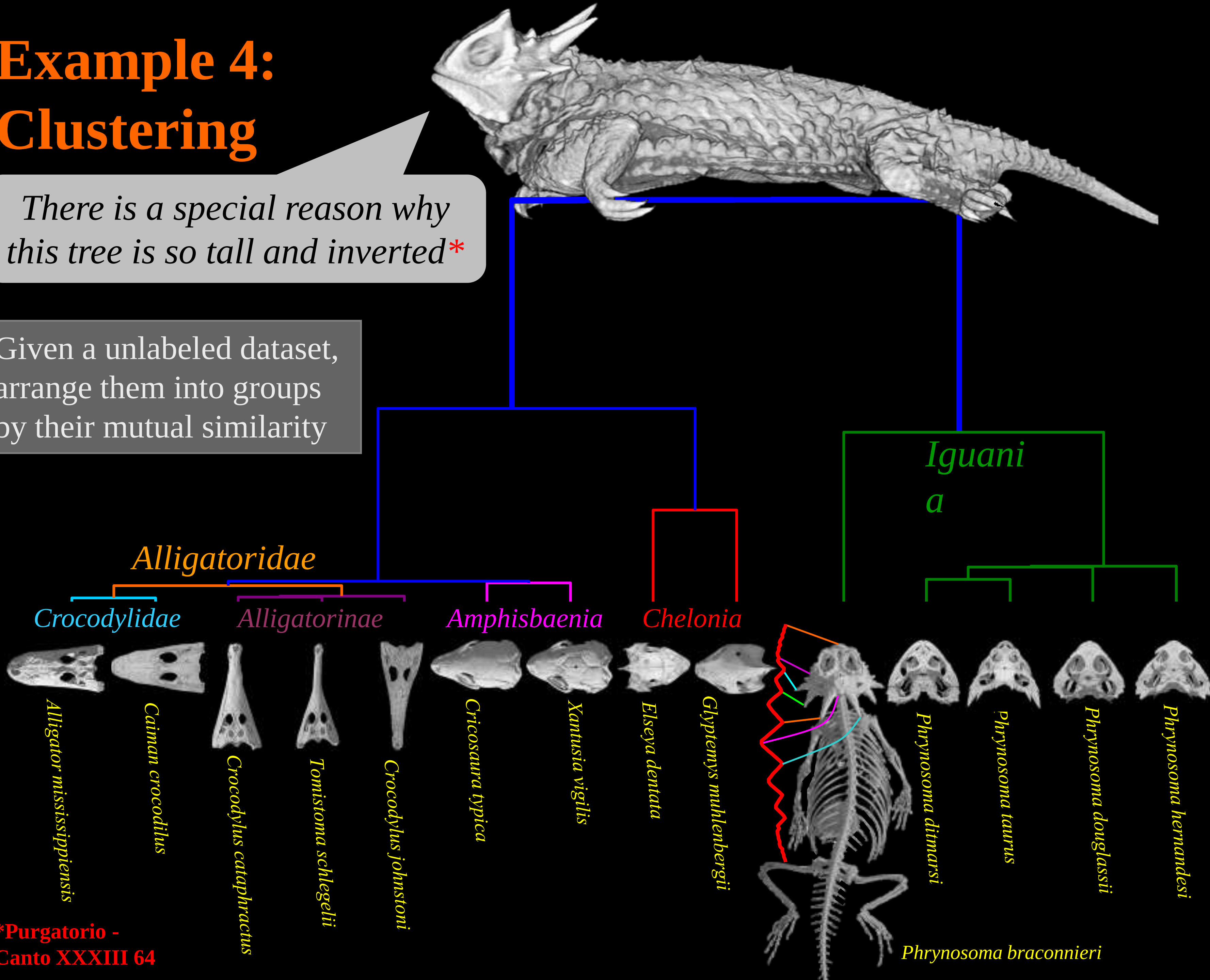


Renegade Canyon 526a
Renegade Canyon 526b
Renegade Canyon 576
Red Rock 1
Sheep Springs 2389
Red Rock 3

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

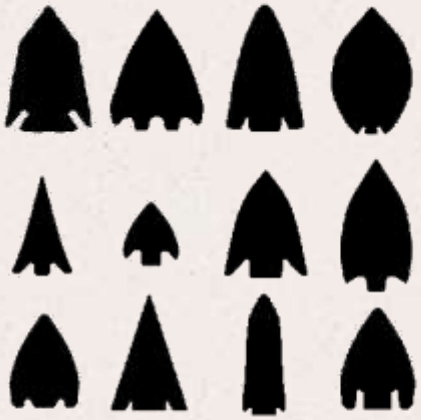


***Purgatorio - Canto XXXIII 64**

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**

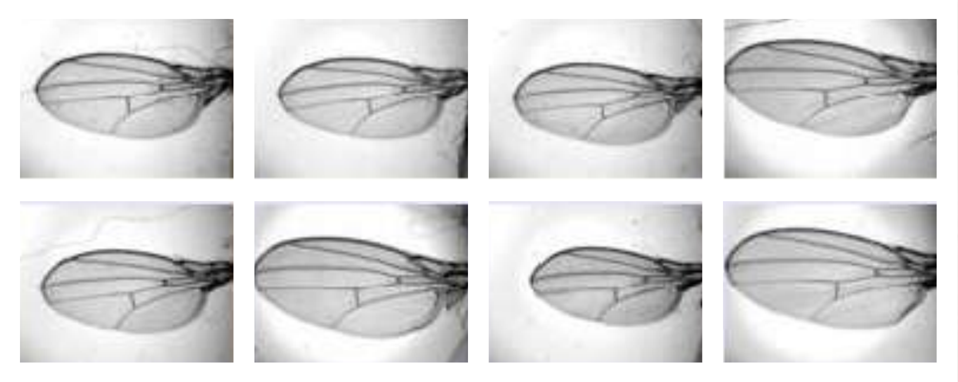
*Paradiso -- Canto XIII 115

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*)

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

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

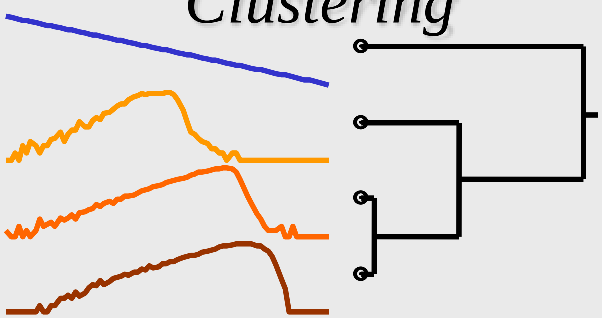
Blythe, California

Baker California

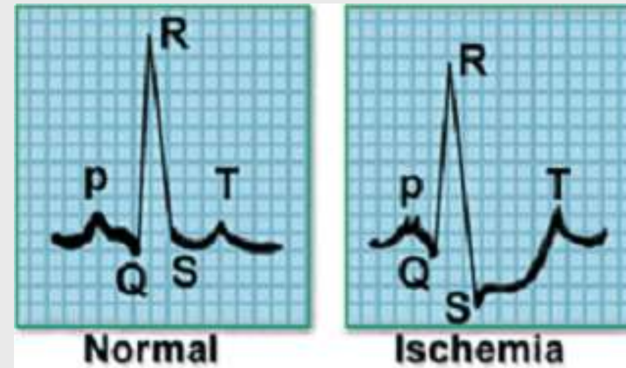
***Inferno -- Canto XIX 15**

All these problems require <u>similarity</u> matching

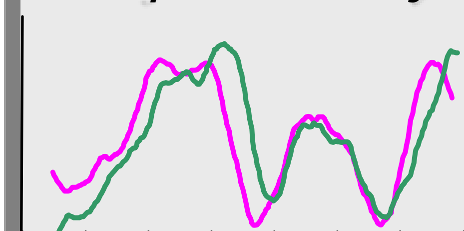
Clustering



Classification



Motif Discovery



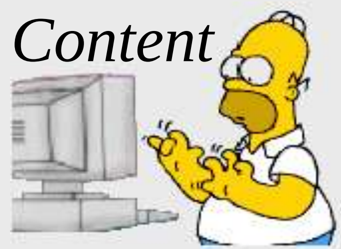
Rule Discovery

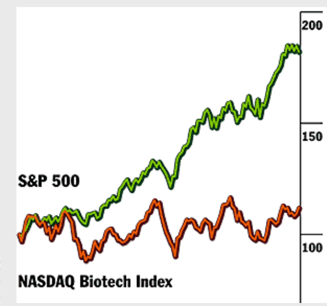
10

$s = 0.5$

$c = 0.3$

Query by Content

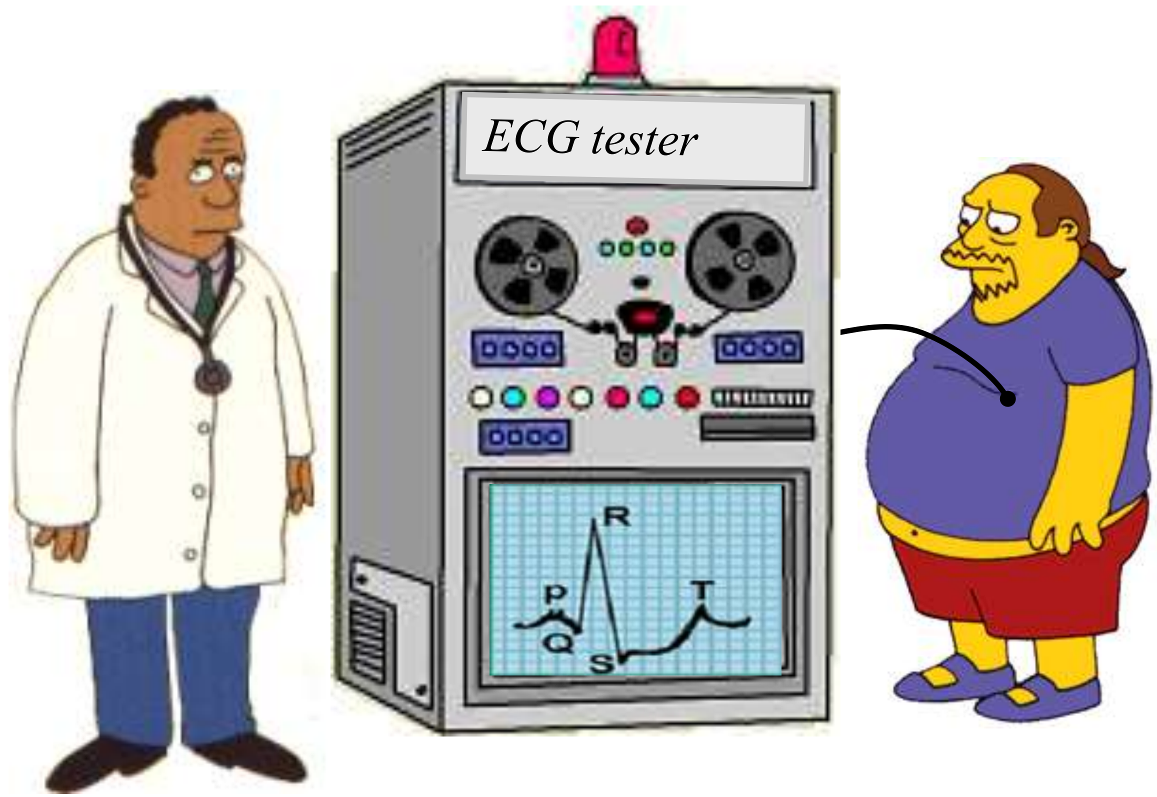




Visualization

Novelty Detection

Here is a simple motivation for the first part of the tutorial



You go to the doctor because of chest pains. Your ECG looks strange…

You doctor wants to search a database to find ***similar*** *ECGs, in the hope that they will offer clues about your condition...*

Two questions:

- ***How do we define similar?***
- *How do we search quickly?*

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features. ***Webster's Dictionary***

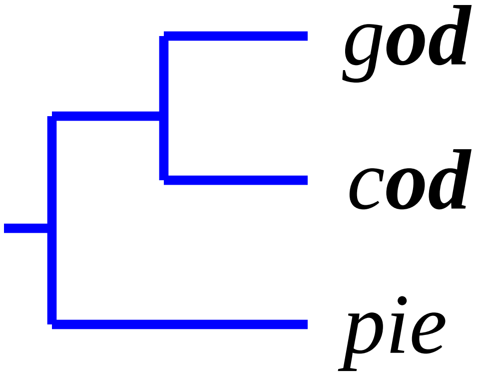
Similarity is hard to define, but…
"We know it when we see it"

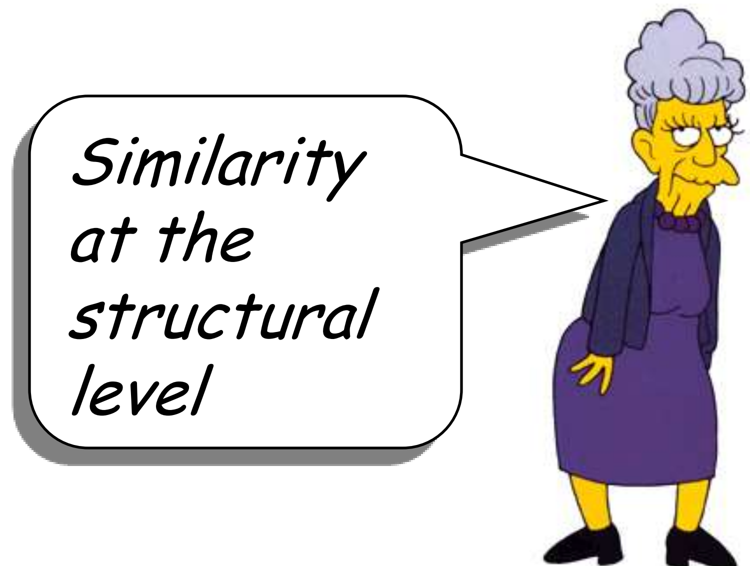
The real meaning of similarity is a philosophical question.

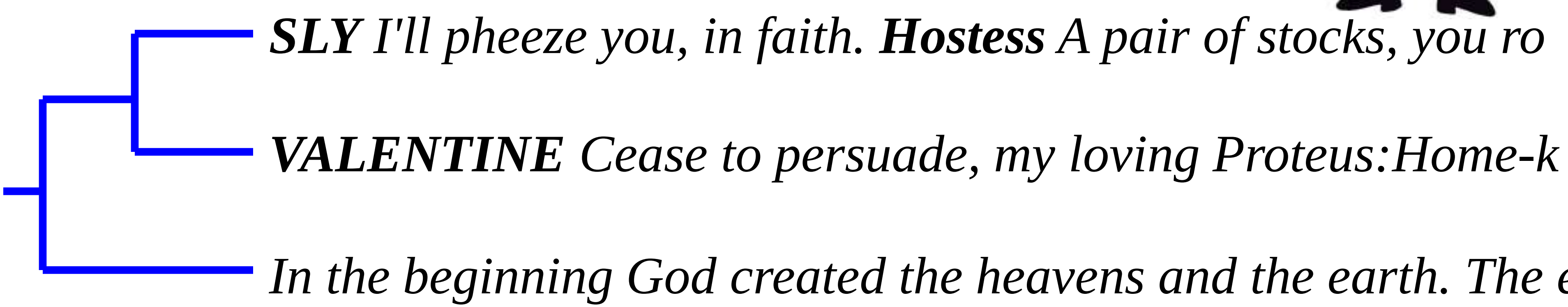
We will take a more pragmatic approach.

Two Kinds of Similarity

text

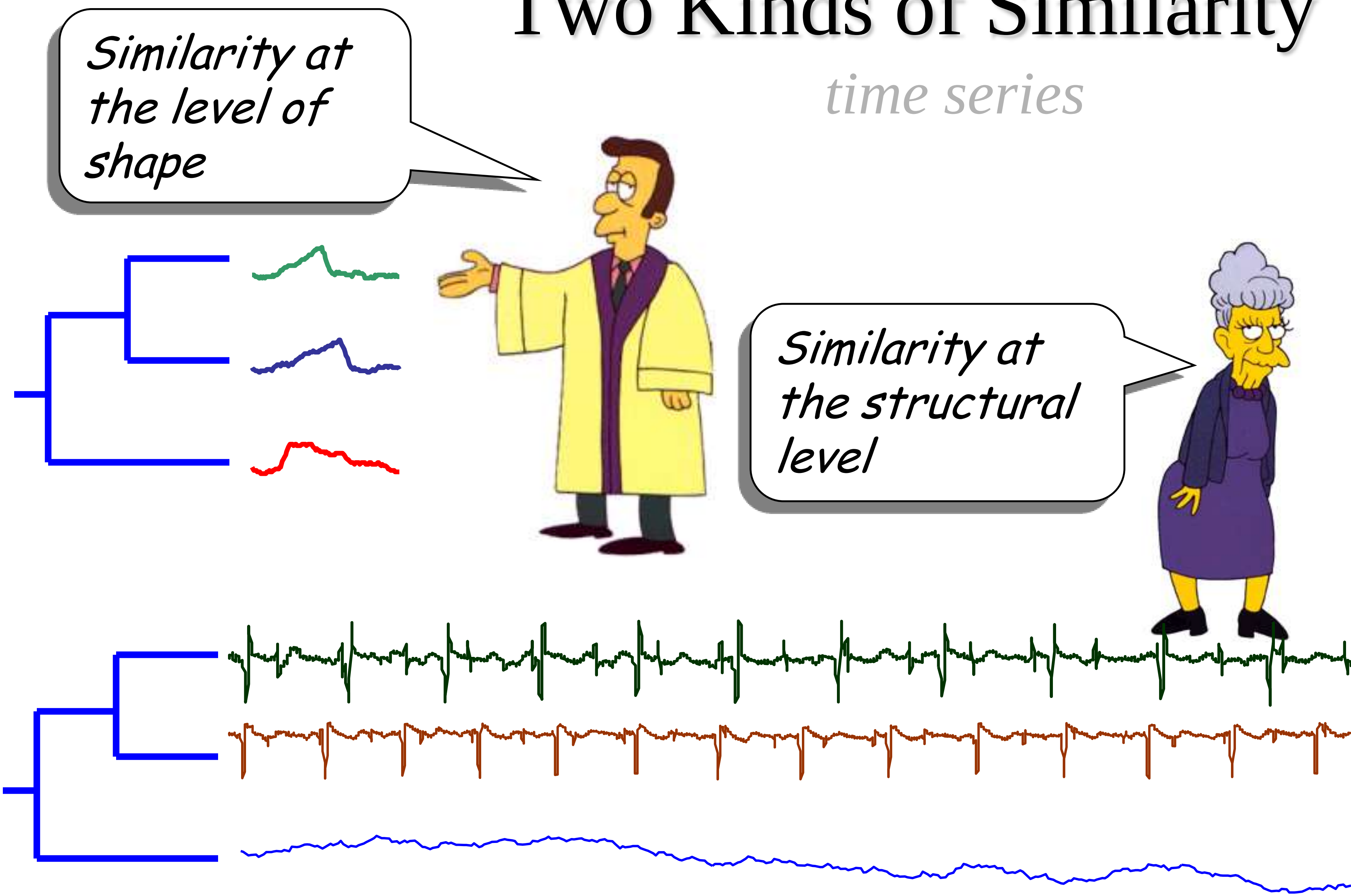






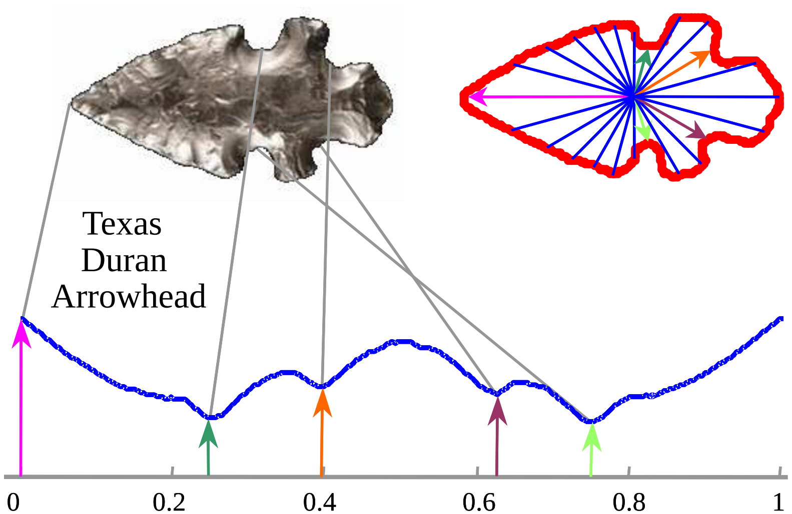
Two Kinds of Similarity

time series



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.

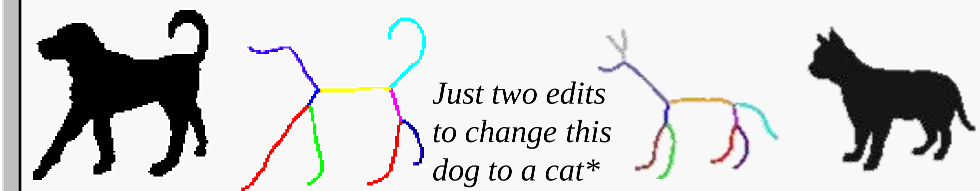
*We **only** consider this approach in this tutorial.*

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*

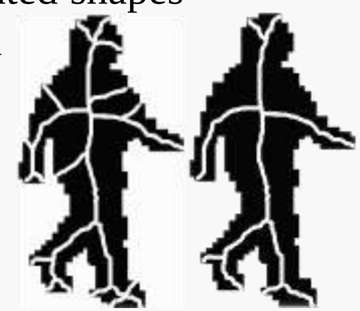
Use graph/tree edit distance to measure similarity



• Some shapes are already "graph like"

• Needed for articulated shapes

• The shape to graph transformation is very tricky[#]

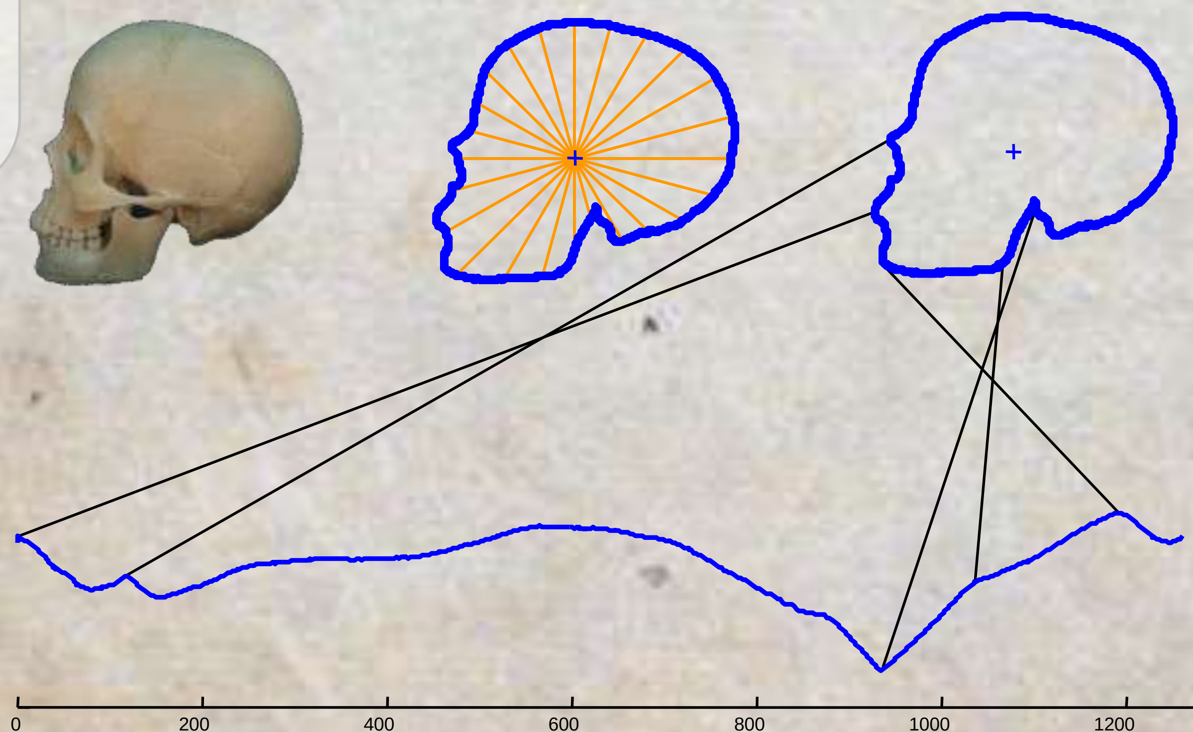


We do not further discuss these ideas, see "shock graph" work of Sebastian, Klein and Kimia* and the work of Latecki[#] and others

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…**

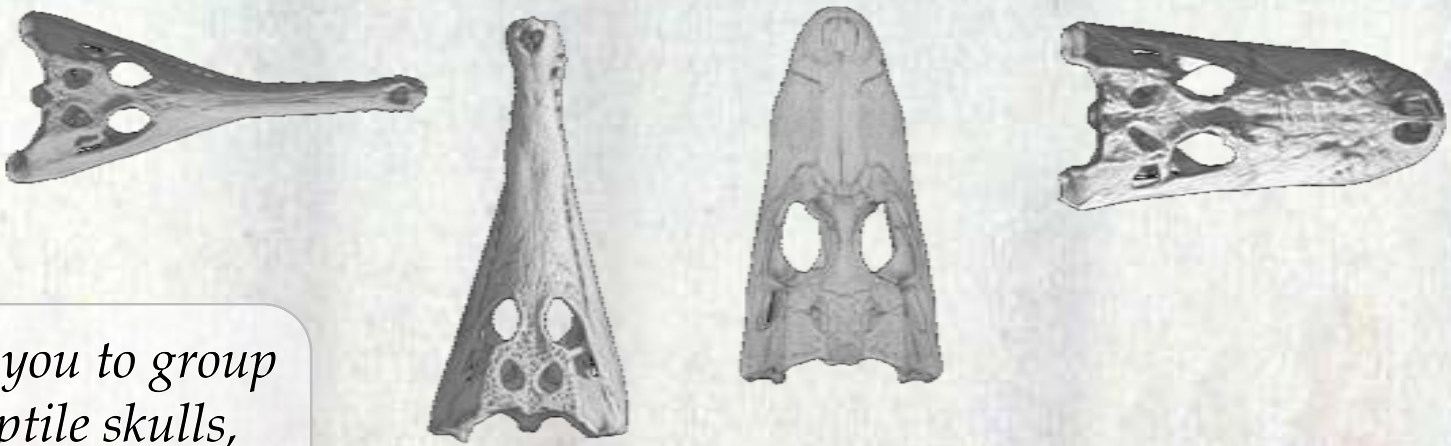


There are many other 1D representations of shape, and the algorithms shown in this tutorial can work with *any* of them

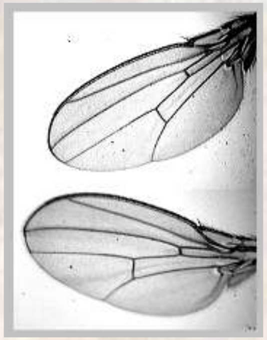
***Paradiso -- Canto XXX, 90.**

Shape Representations

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



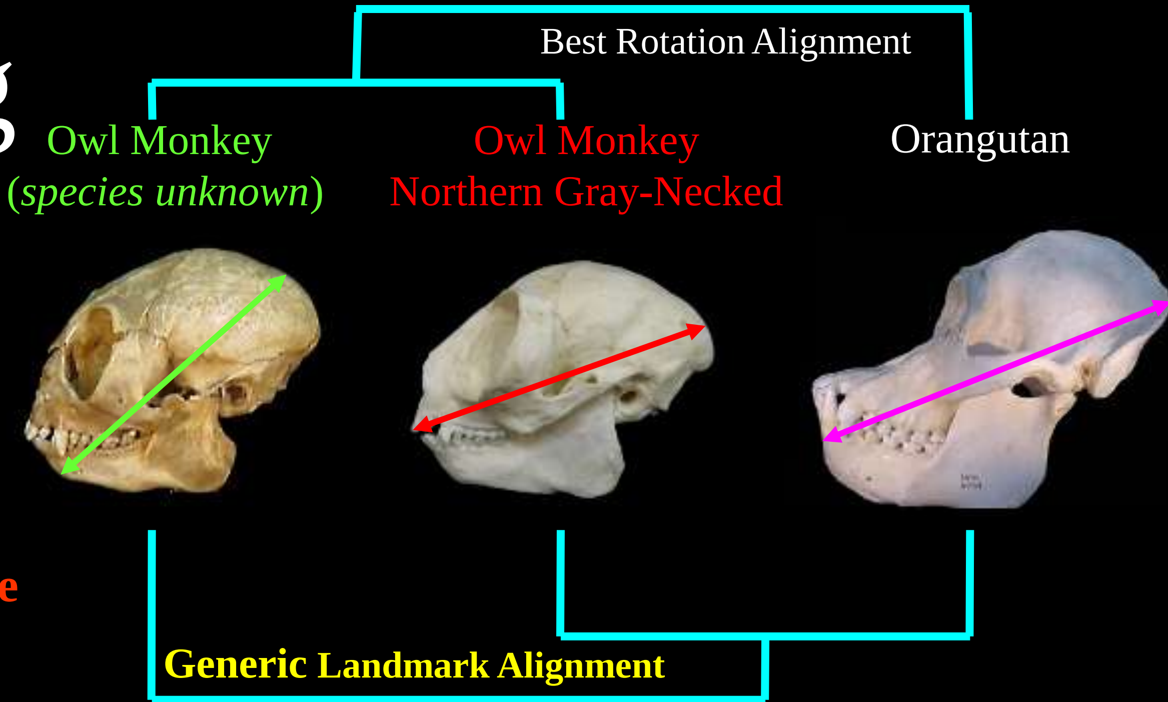
If I asked you to group these reptile skulls, rotation would not confuse you



There are two ways to be rotation invariant

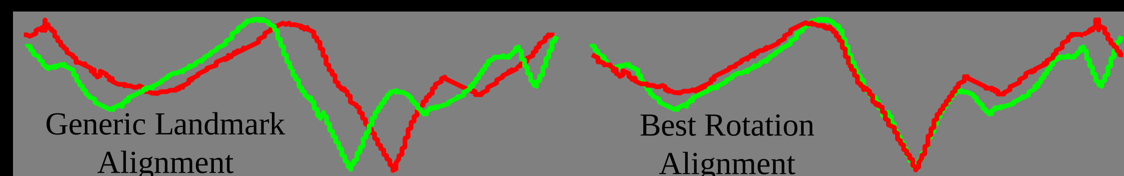
1) Landmarking: Find the one "true" rotation
2) Rotation invariant features

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 …



The only problem with landmarking is that it does not work

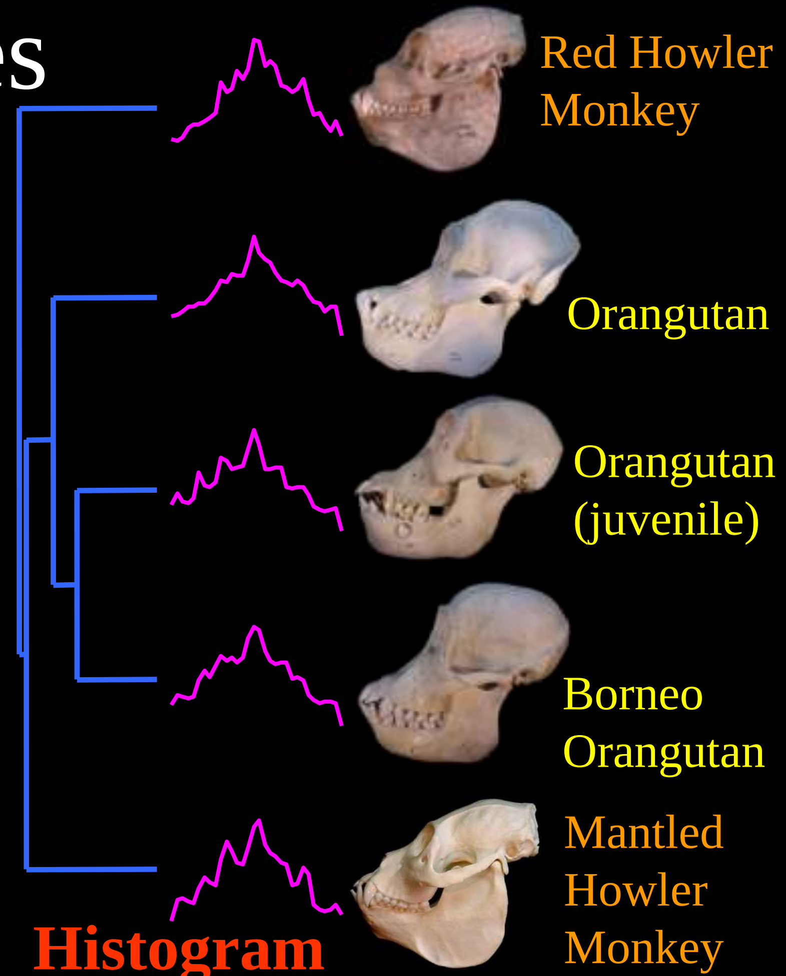
Domain Specific Landmarking

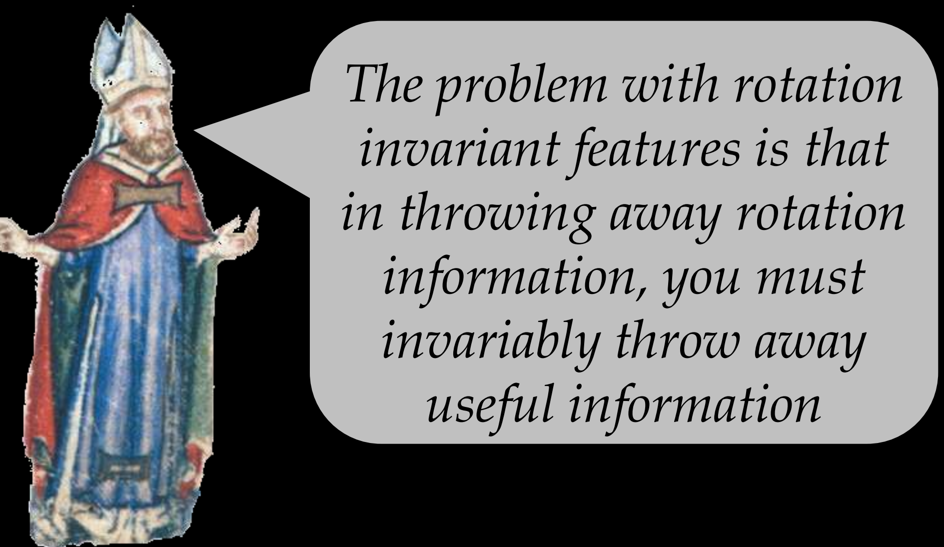
Domain specific landmarks include leaf stems, noses, the tip of arrowheads…

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**



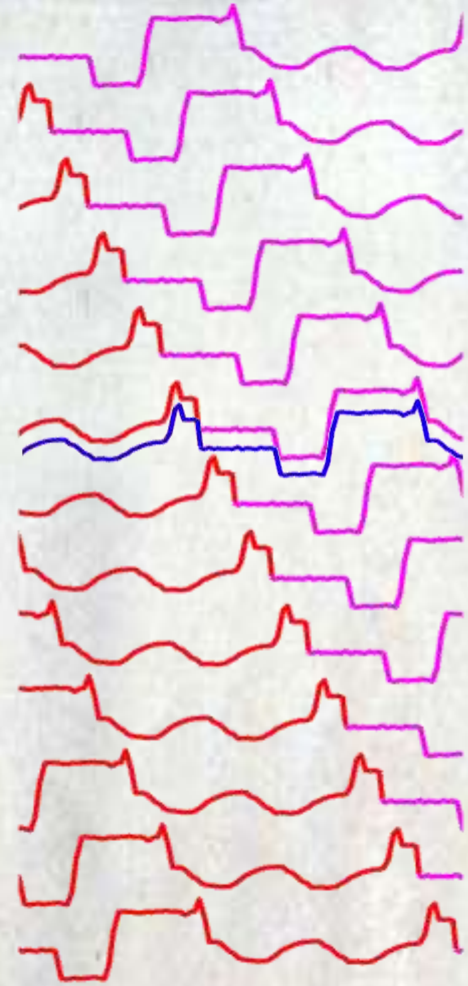


aspect ratio (monkeys)

aspect ratio (reptiles)

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]

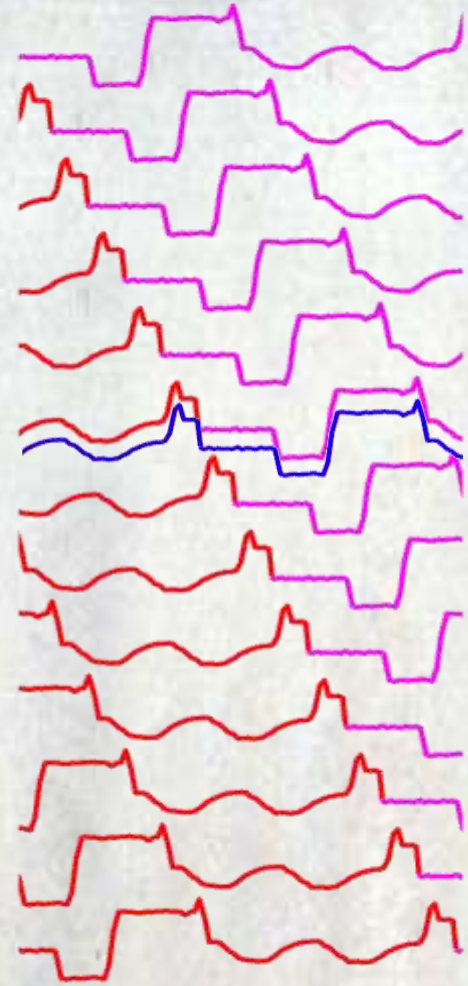


$$\boldsymbol{C} = \begin{Bmatrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{Bmatrix}$$

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

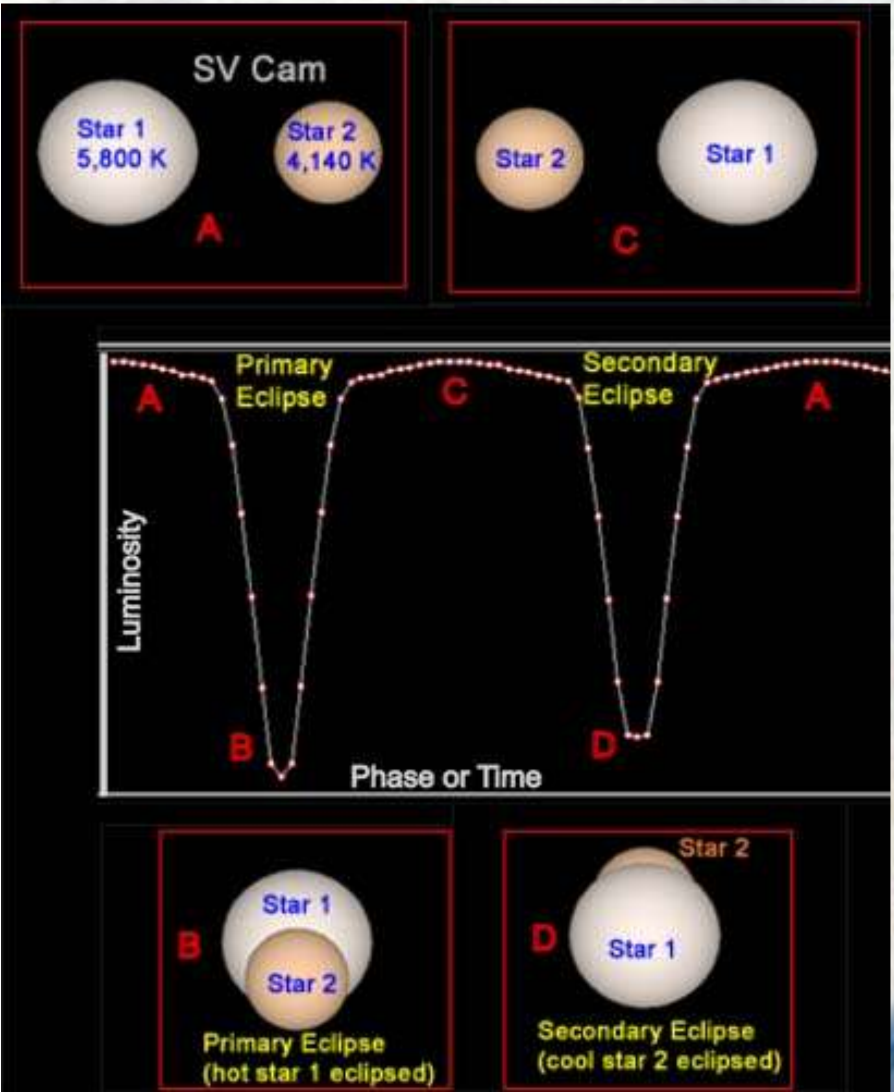
*However there now exists a simple **exact** ultrafast, indexable way to do this**



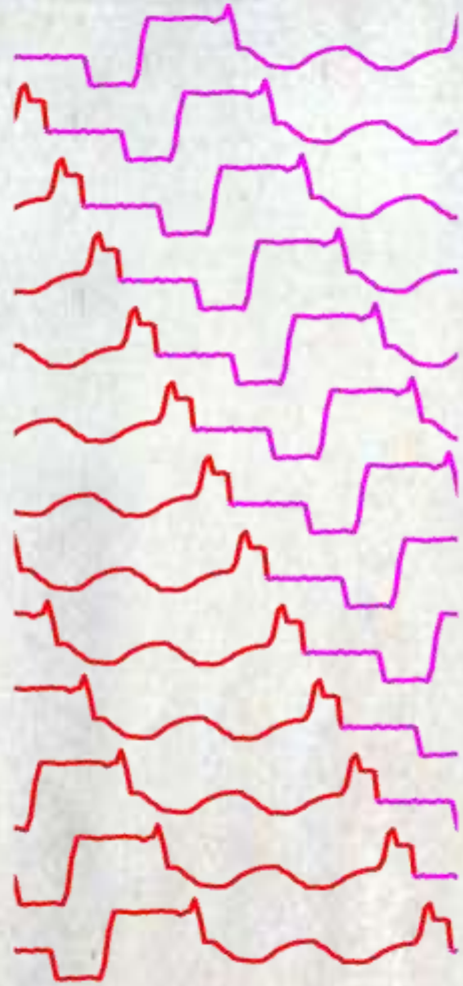
$$\mathbf{C} = \begin{Bmatrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{Bmatrix}$$

***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



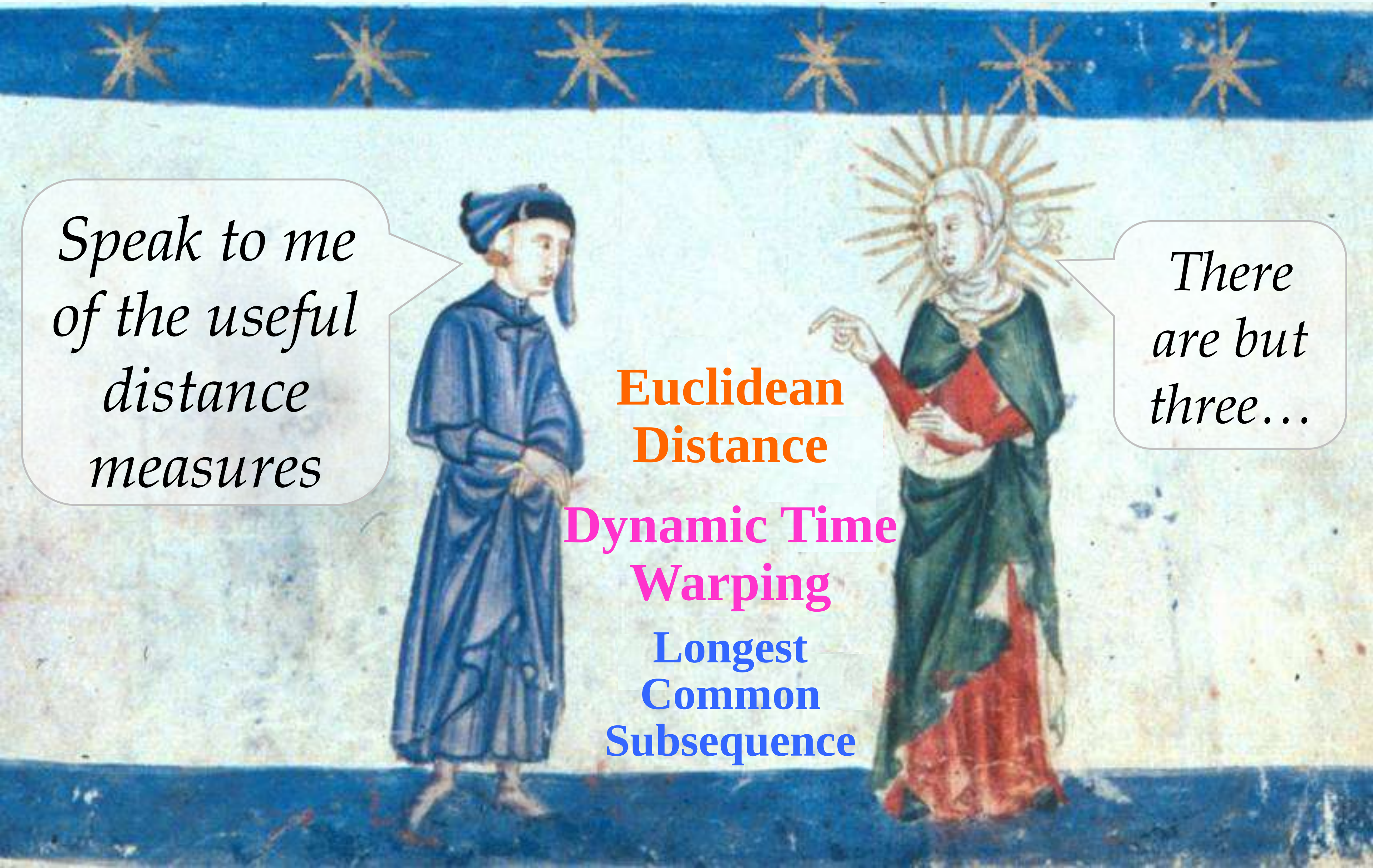
I saw above a million burning lamps,
*A Sun kindled every one of them, as our sun lights the stars we glimpse on high**



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

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

Shape Distance Measures



Defining Distance Measures

Definition: *Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) is denoted by $D(O_1,O_2)$*

What properties are desirable in a distance measure?

• $D(A,B) = D(B,A)$	*Symmetry*
• $D(A,A) = 0$	*Constancy*
• $D(A,B) = 0$ iff $A = B$	*Positivity*
• $D(A,B) \leq D(A,C) + D(B,C)$	*Triangular Inequality*

Intuitions behind desirable distance measure properties I

$D(A,B) = D(B,A)$ *Symmetry*

D( , ) = D( , )

Otherwise you could claim:

Patty looks like Selma, but Selma does not look like Patty!

Intuitions behind desirable distance measure properties II

D(A,A) = 0 *Constancy of Self-Similarity*

D(*,**) = 0*

Otherwise you could claim:

Intuitions behind desirable distance measure properties III

D(A,B) = 0, IIf A=B *Positivity*

D( *,*  *) = 0, IFF*  *=* 

Otherwise you could claim:

I know Patty and Marge are somehow different, but I can't tell them apart!

Intuitions behind desirable distance measure properties IV

$D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

D( , ) ≤ D( , ) + D( , )

Otherwise you could claim:

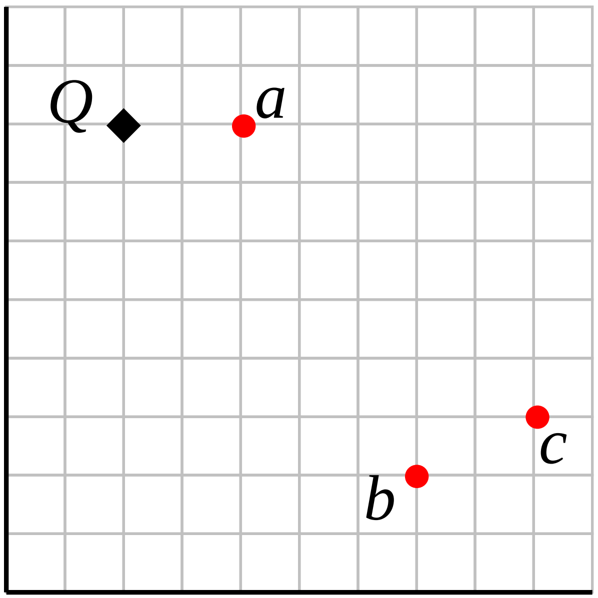
Patty looks like Marge, Selma also looks like Marge, But Patty looks nothing like Selma!

Why is the Triangular Inequality so Important?

Virtually all techniques to index data require the triangular inequality to hold.

Suppose I am looking for the closest point to Q, in a database of 3 objects.

Further suppose that the triangular inequality holds, and that we have precomplied a table of distance between all the items in the database.



	a	b	c
a		6.70	7.07
b			2.30
c			

Why is the Triangular Inequality so Important?

Virtually all techniques to index data require the triangular inequality to hold.

*I find **a** and calculate that it is 2 units from Q, it becomes my best-so-far. I find **b** and calculate that it is* 7.81 *units away from Q.*

*I don't have to calculate the distance from Q to **c**!*

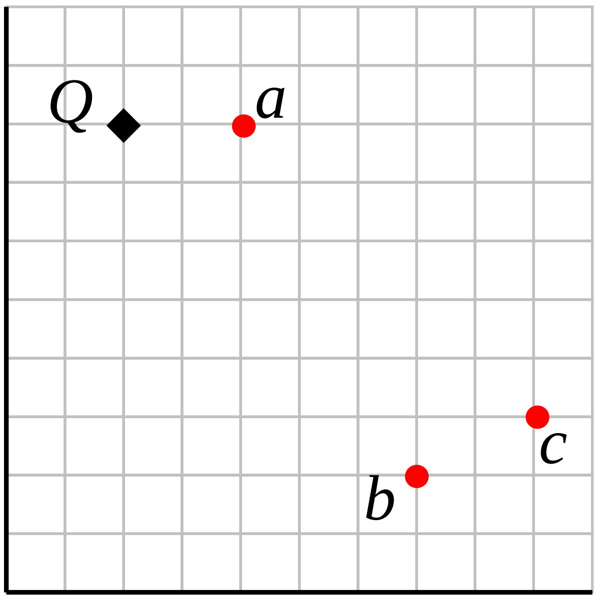
I know $D(Q,\mathbf{b}) \leq D(Q,\mathbf{c}) + D(\mathbf{b},\mathbf{c})$

$$D(Q,\mathbf{b}) - D(b,\mathbf{c}) \leq D(Q,\mathbf{c})$$

$$7.81 - 2.30 \leq D(Q,\mathbf{c})$$

$$5.51 \leq D(Q,\mathbf{c})$$

*So I know that **c** is at least 5.51 units away, but my best-so-far is only 2 units away.*

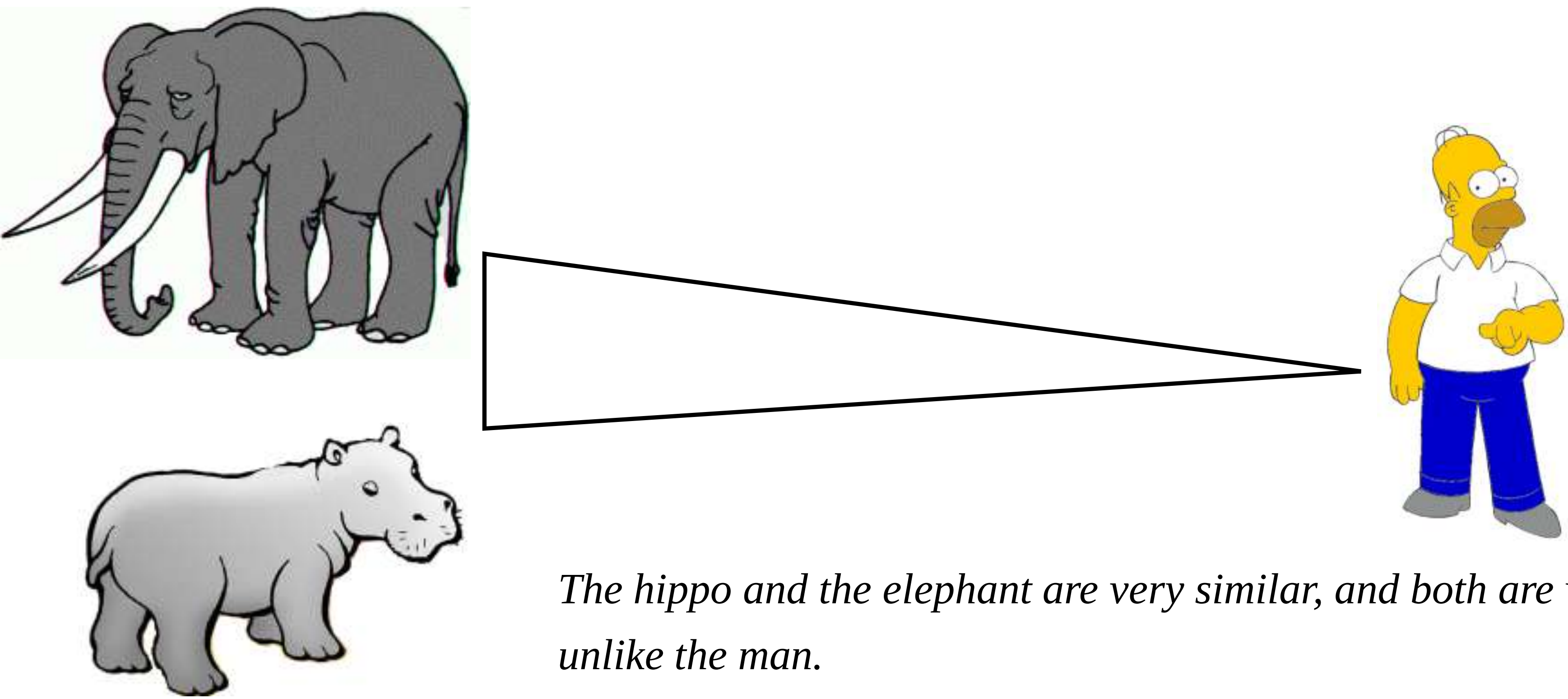


	a	b	c
a		6.70	7.07
b			**2.30**
c			

A Final Thought on the Triangular Inequality I

Sometimes the triangular inequality requirement maps nicely onto human intuitions.

Consider the similarity between a hippo, an elephant and a man.

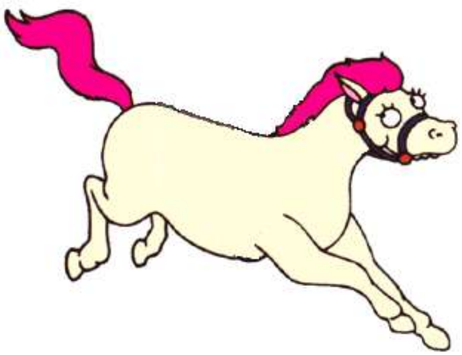


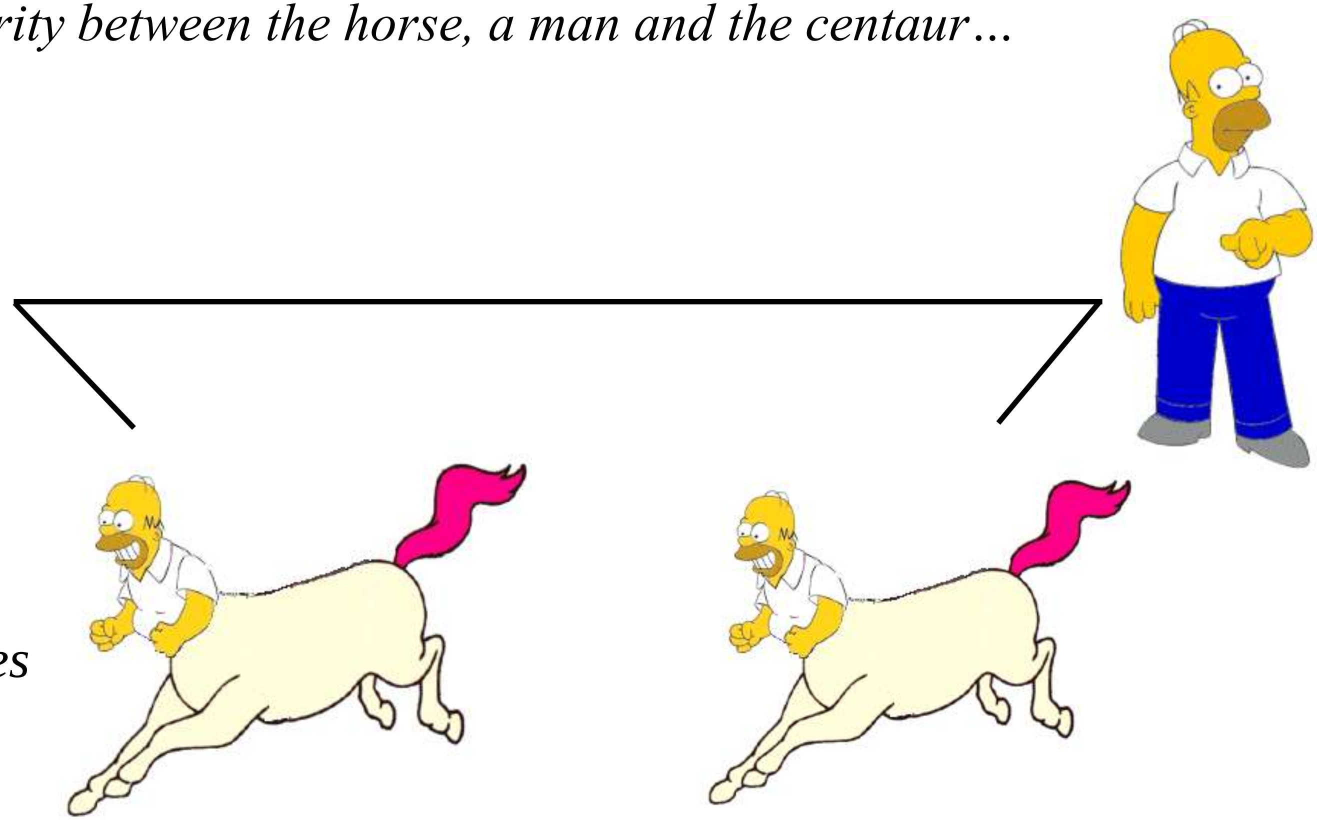
The hippo and the elephant are very similar, and both are very unlike the man.

A Final Thought on the Triangular Inequality II

Sometimes the triangular inequality requirement fails to map onto human intuition.

Consider the similarity between the horse, a man and the centaur…





The ***horse*** *and the* ***man*** *are very different, but both share many features with the* ***centaur****.*
This relationship does not obey the triangular inequality.

This example due to Remco C. Veltkamp

Preprocessing the data before distance calculations

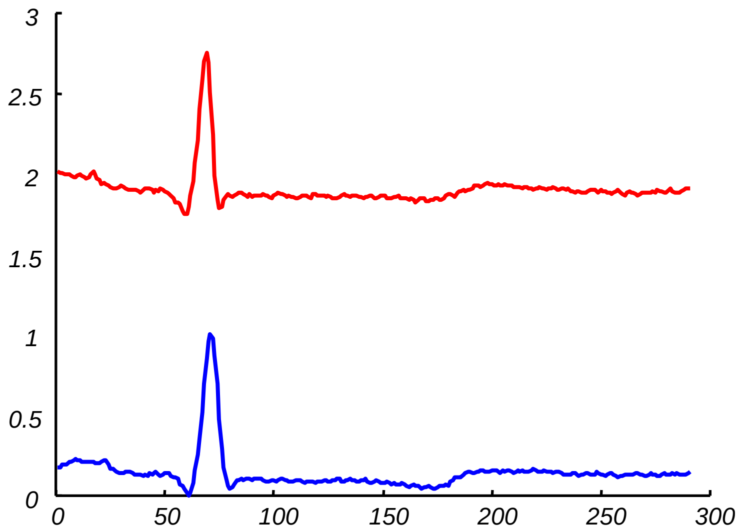
If we naively try to measure the distance between two "raw" time series, we may get very unintuitive results

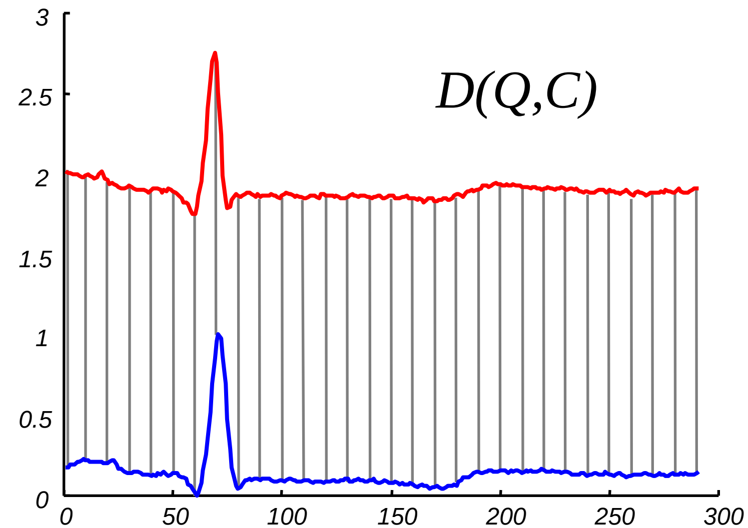
This is because Euclidean distance is very sensitive to some "distortions" in the data. For most problems these distortions are not meaningful, and thus we can and should remove them

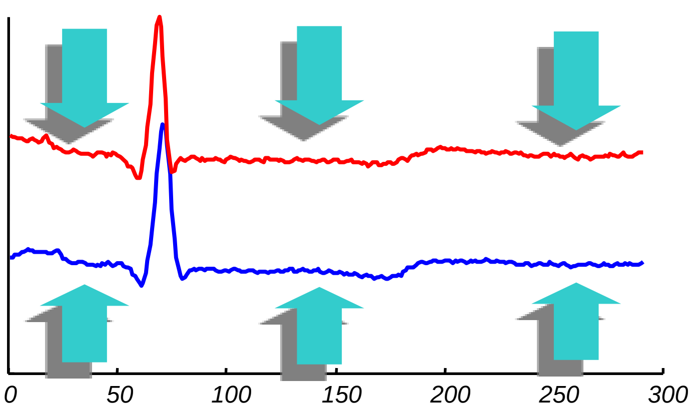
In the next few slides we will discuss the 4 most common distortions, and how to remove them

- *Offset Translation*
- *Amplitude Scaling*
- *Linear Trend*
- *Noise*

Transformation I: Offset Translation



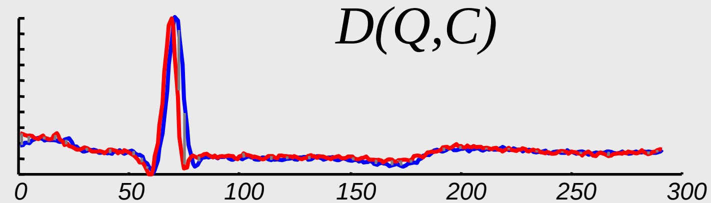




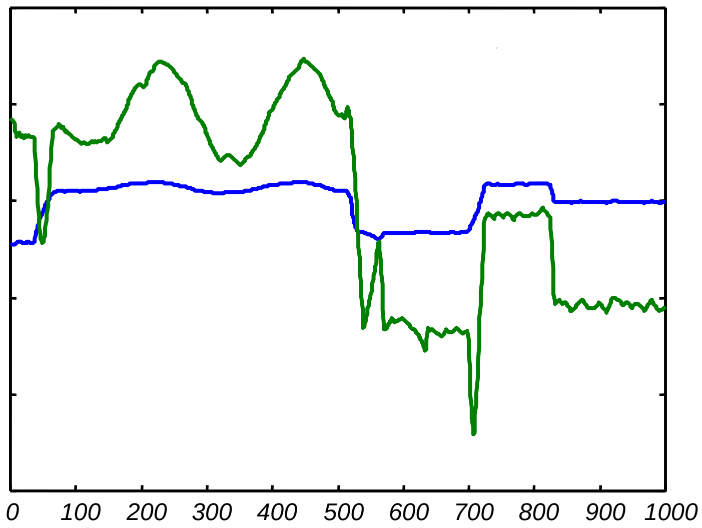
$Q = Q - mean(Q)$

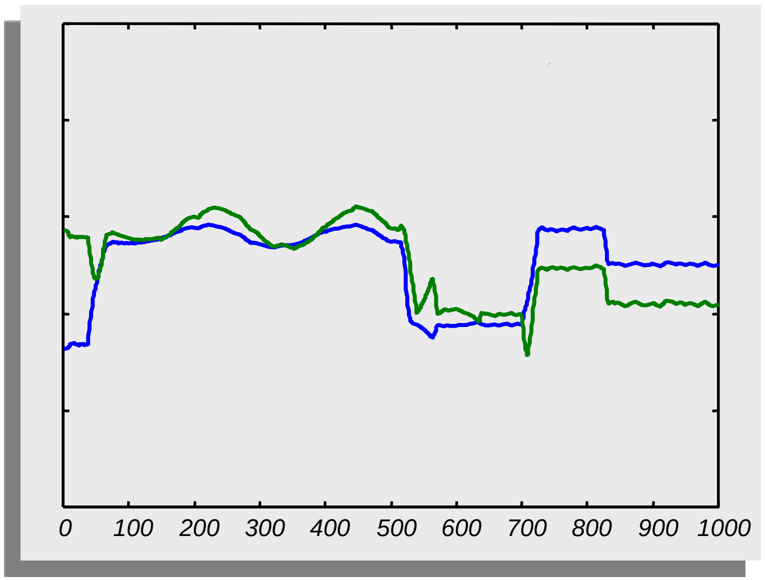
$C = C - mean(C)$

$D(Q,C)$



Transformation II: Amplitude Scaling





For fast normalization, see:

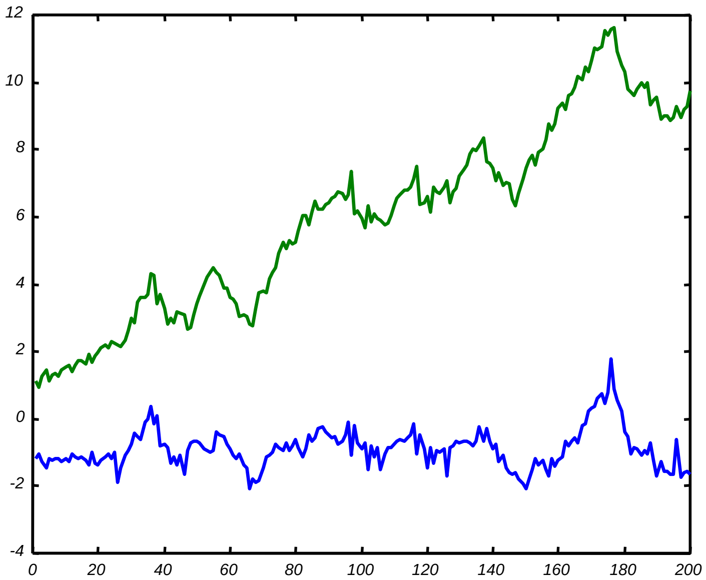
Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K.(1995). Fast similarity search in the presence of noise, scaling, and translation in times-series databases. In VLDB, September.

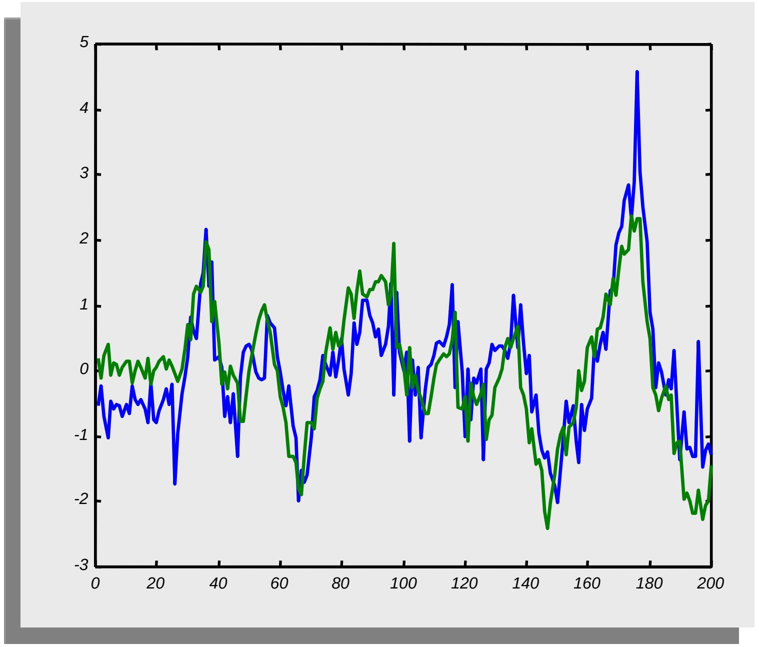
Q = (Q - mean(Q)) / std(Q)

C = (C - mean(C)) / std(C)

D(Q,C)

Transformation III: Linear Trend





The intuition behind removing linear trend is…

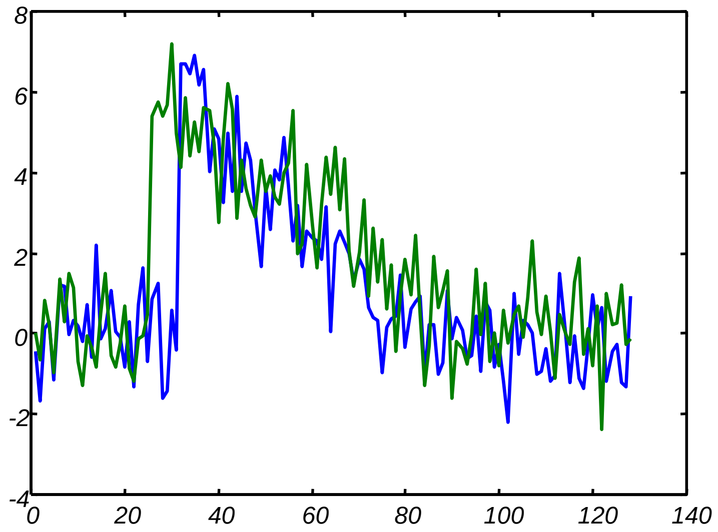
Fit the best fitting straight line to the time series, then subtract that line from the time series.

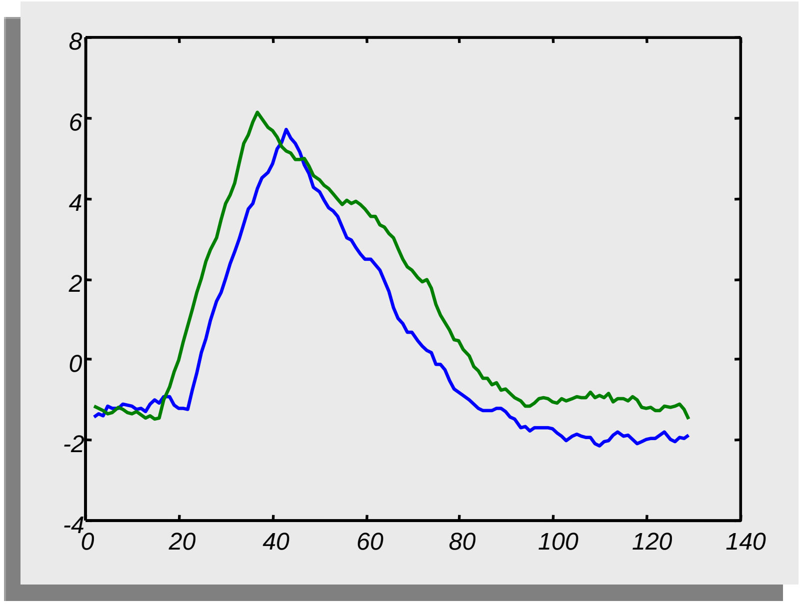
Removed ***linear trend***

Removed offset translation

Removed amplitude scaling

Transformation IIII: Noise





The intuition behind removing noise is...

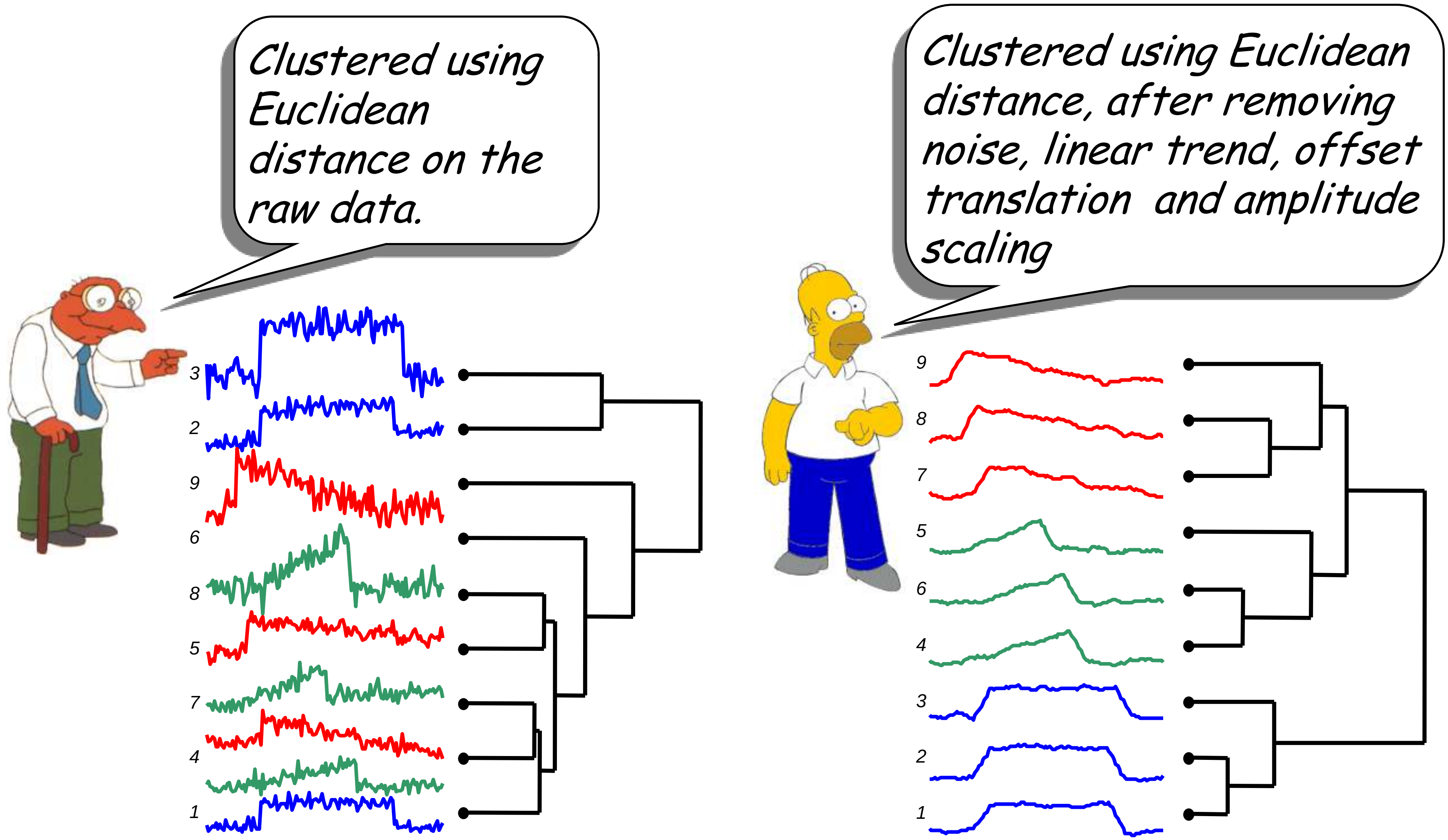
Average each datapoint's value with its neighbors.

Q = smooth(Q)

C = smooth(C)

D(Q,C)

A Quick Experiment to Demonstrate the Utility of Preprocessing the Data



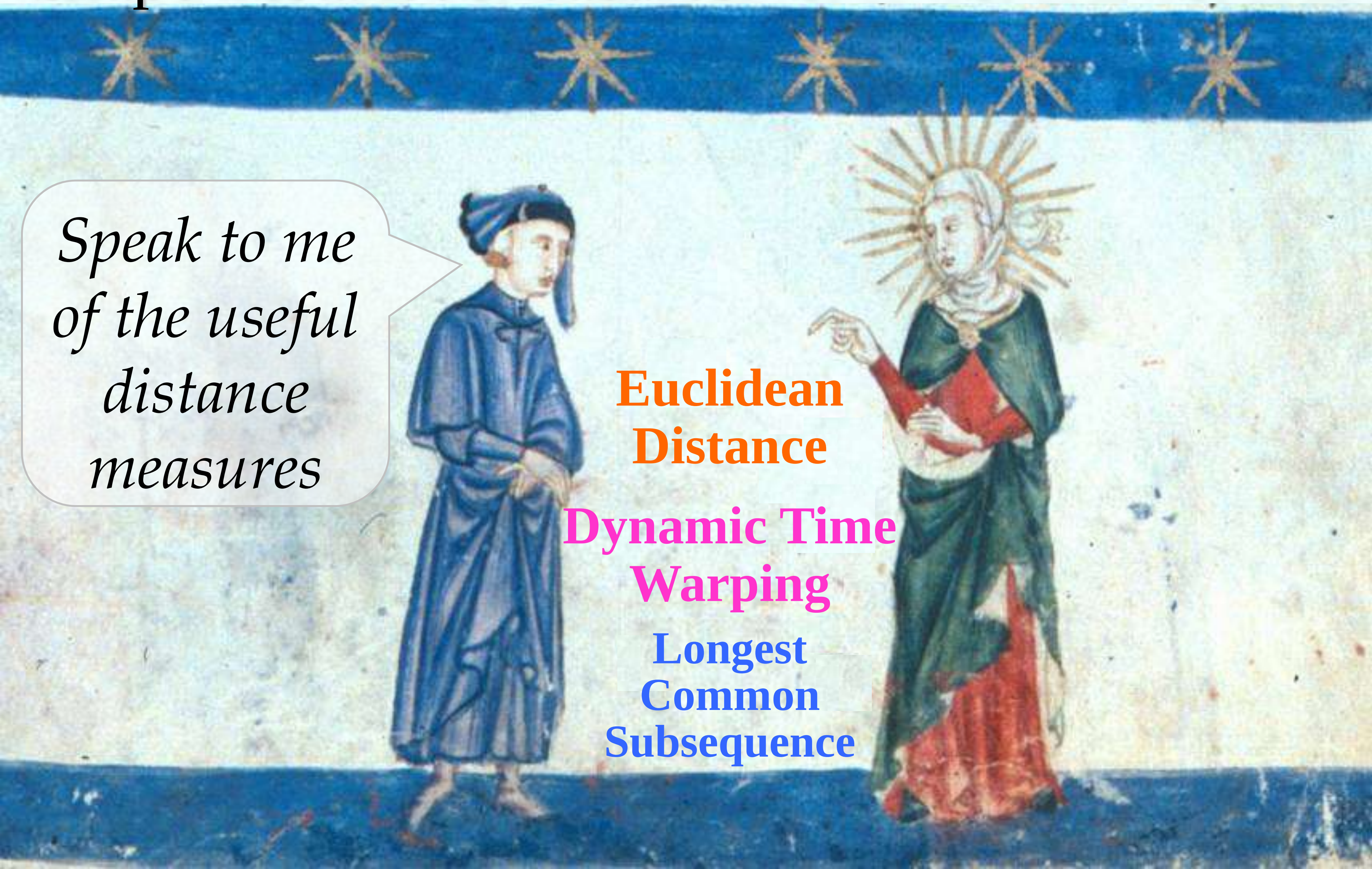
Summary of Preprocessing

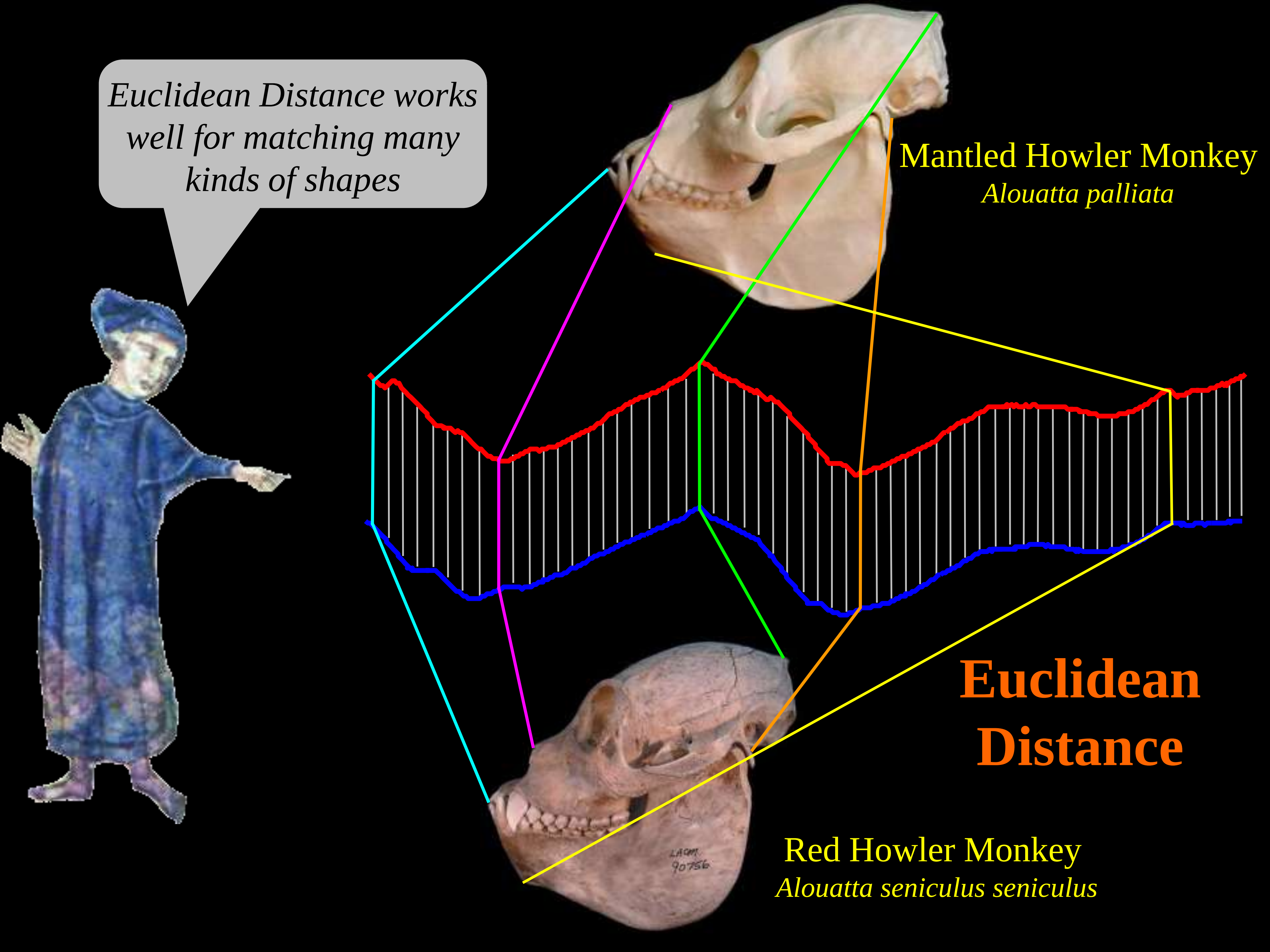
The "raw" time series may have distortions which we should remove before clustering, classification etc

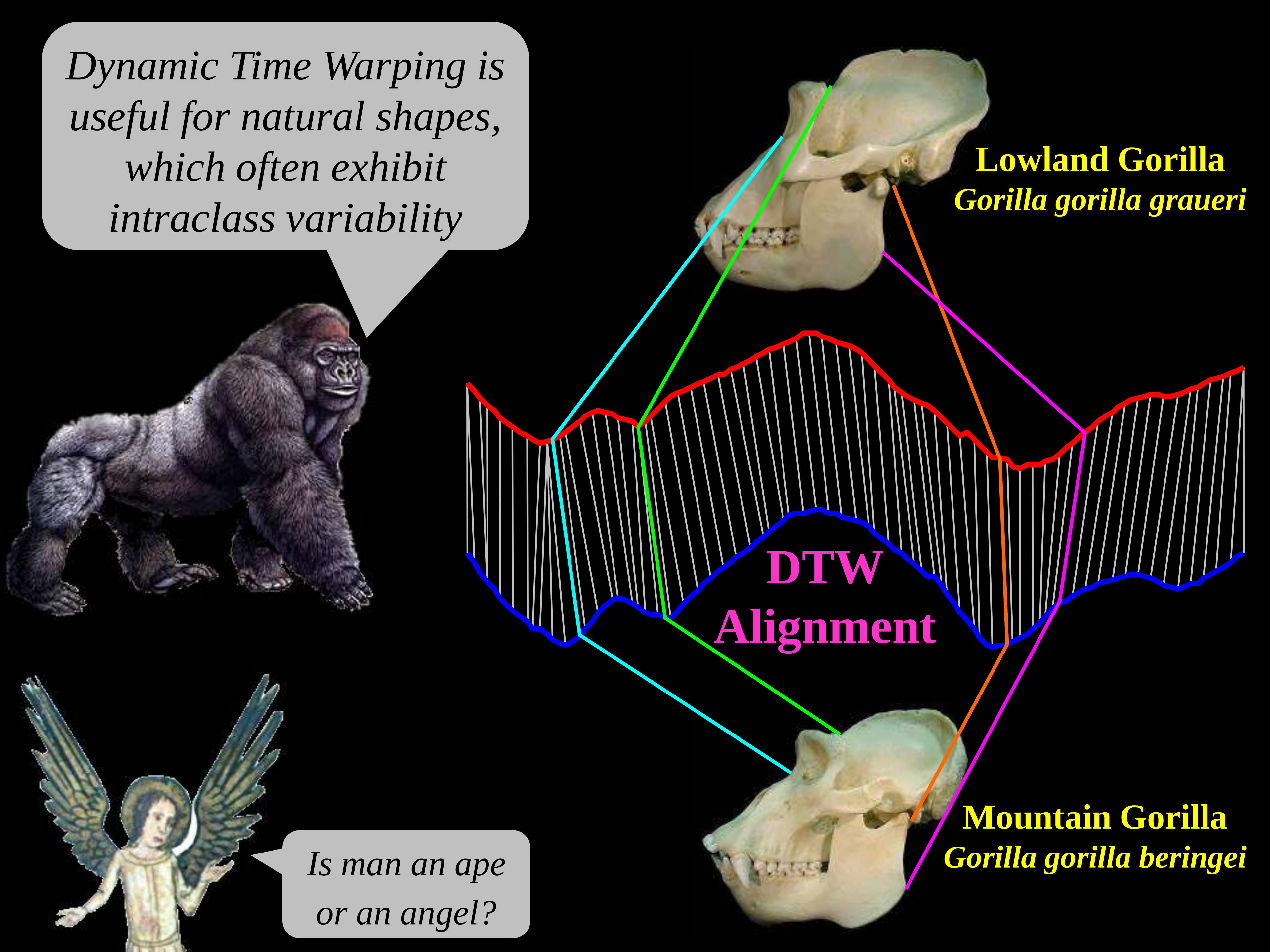
Of course, sometimes the distortions are the most interesting thing about the data, the above is only a general rule

We should keep in mind these problems as we consider the high level representations of time series which we will encounter later (DFT, Wavelets etc). Since these representations often allow us to handle distortions in elegant ways

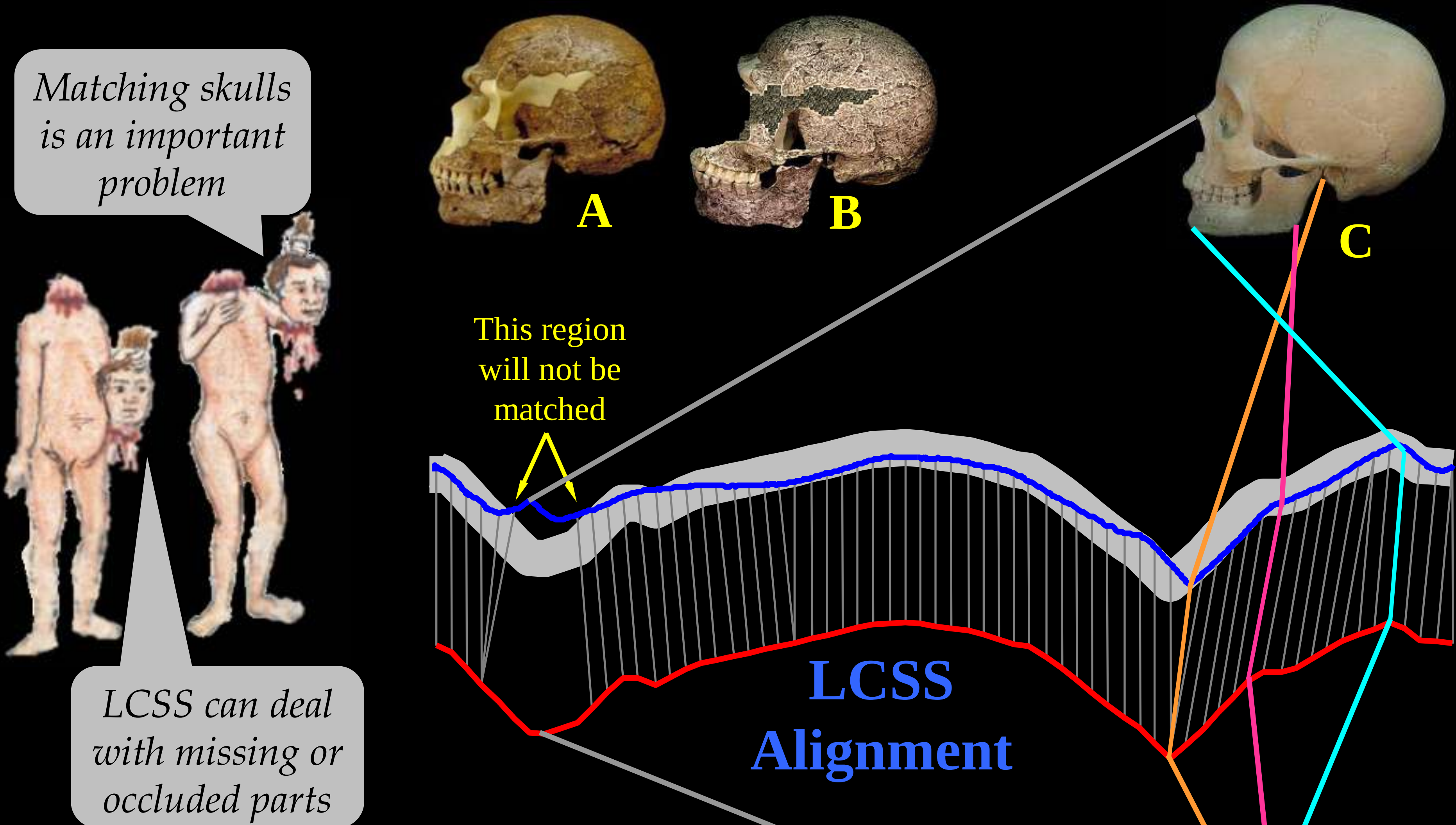
Back to
Shape Distance Measures

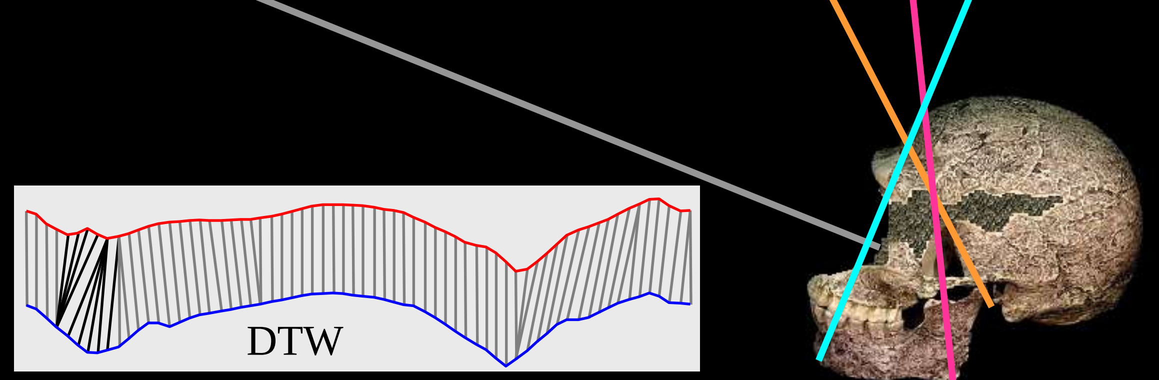



Euclidean Distance works well for matching many kinds of shapes
Mantled Howler Monkey
Alouatta palliata
Euclidean Distance
Red Howler Monkey
Alouatta seniculus seniculus
LACM 90756



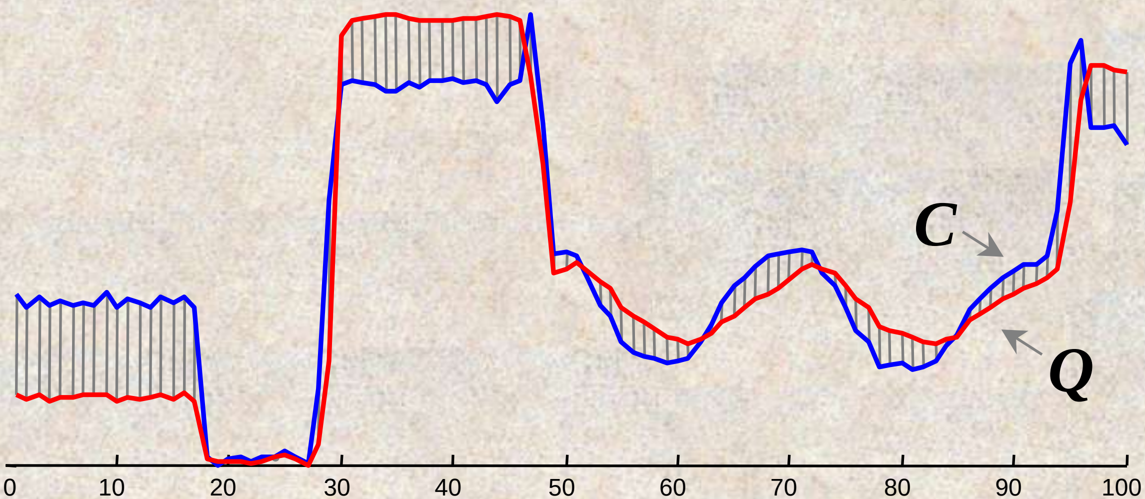
Dynamic Time Warping is useful for natural shapes, which often exhibit intraclass variability
Lowland Gorilla
Gorilla gorilla graueri
DTW Alignment
Mountain Gorilla
Gorilla gorilla beringei
Is man an ape or an angel?





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



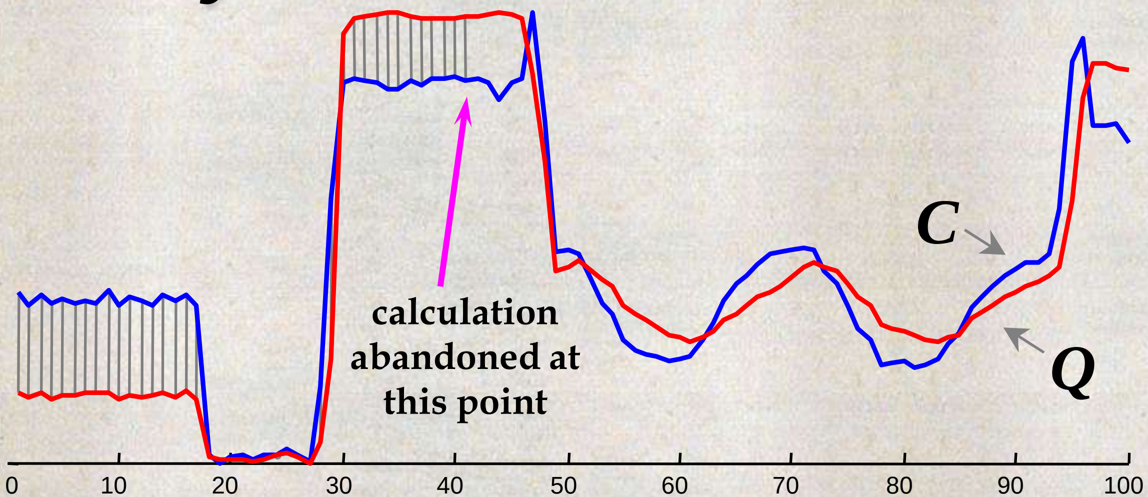
Given two time series $Q = q_1 \ldots q_n$ and $C = c_1 \ldots c_n$, the Euclidean distance between them is defined as:

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

I notice that you Z-normalized the time series first

The next slide shows a useful optimization…

Early Abandon Euclidean Distance

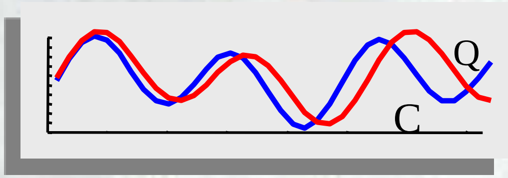


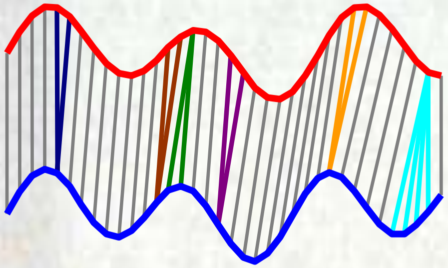
During the computation, if current sum of the squared differences between each pair of corresponding data points exceeds r^2, we can safely ***abandon*** *the calculation*

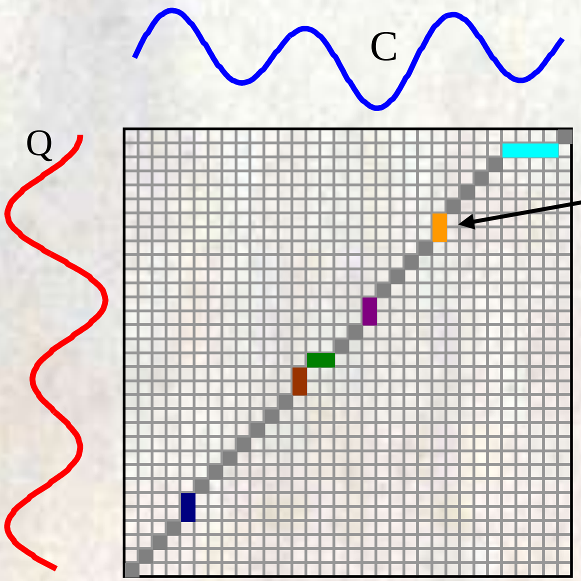
I see, because incremental value is always a lower bound to the final value, once it is greater than the best-so-far, we may as well abandon

Abandon all hope ye who enter here

Dynamic Time Warping I







Warping path *w*

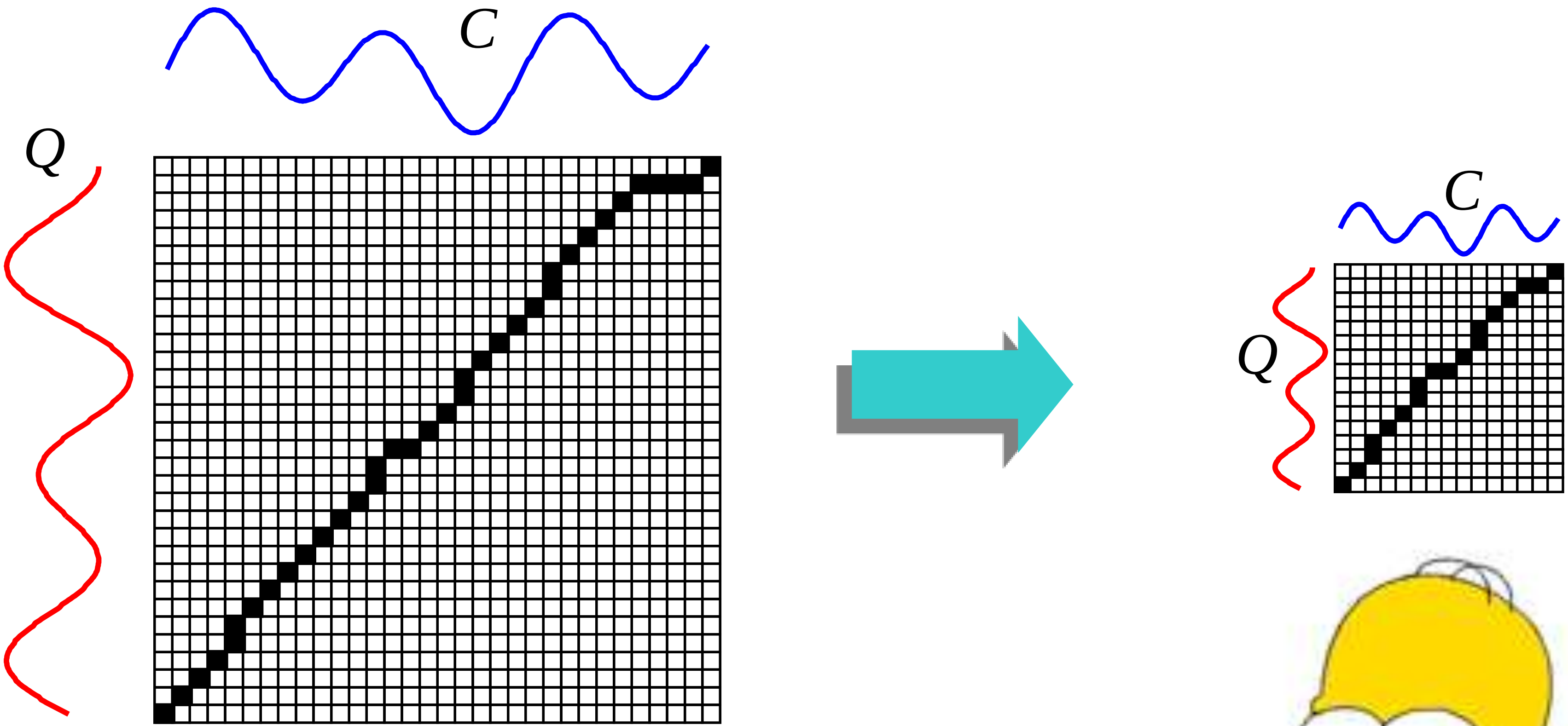
This is how the DTW alignment is found

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

This recursive function gives us the minimum cost path

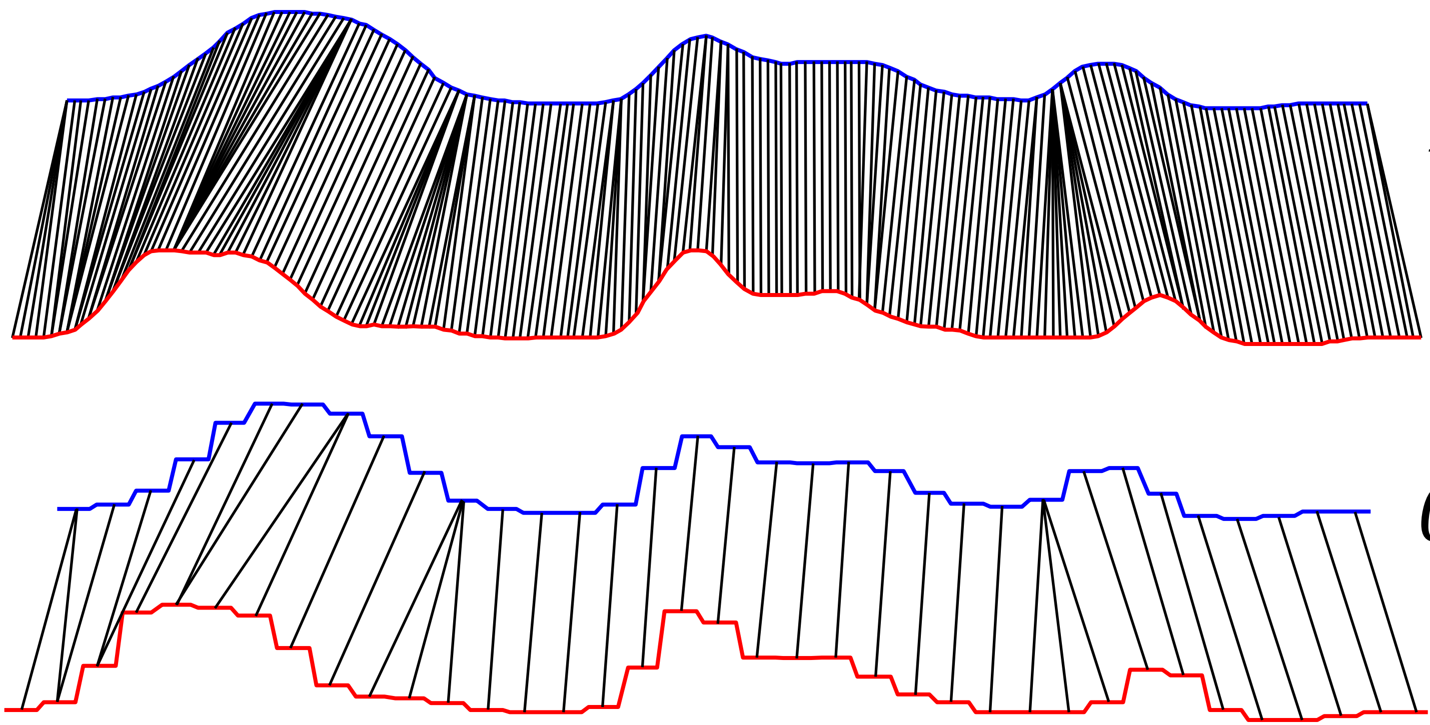
$$\gamma(\mathrm{i,j}) = d(q_{\mathrm{i}},c_{\mathrm{j}}) + \min\{ \gamma(i\text{-}1,j\text{-}1), \gamma(i\text{-}1,j), \gamma(i,j\text{-}1) \}$$

Fast Approximations to Dynamic Time Warp Distance I



Simple Idea: Approximate the time series with some compressed or downsampled representation, and do DTW on the new representation. How well does this work...

Fast Approximations to Dynamic Time Warp Distance II



1.03 sec

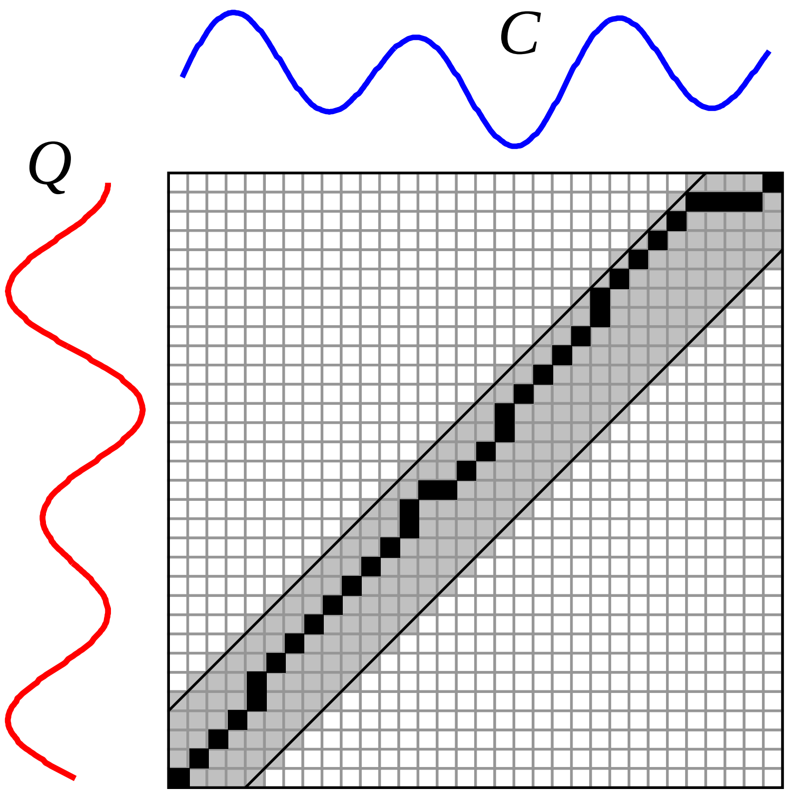
0.07 sec

… there is strong visual evidence to suggests it works well

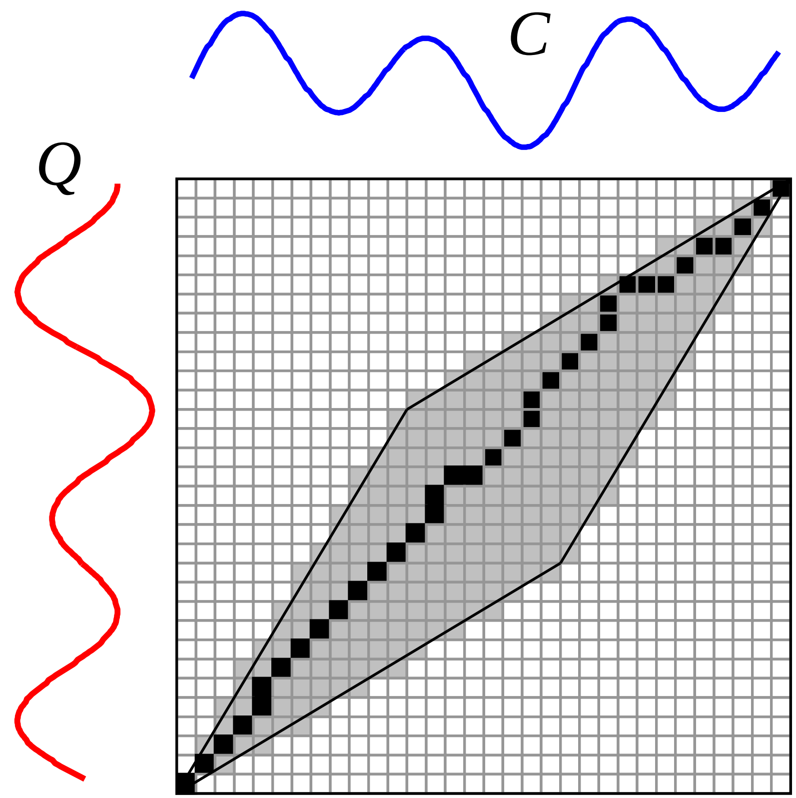
There is good experimental evidence for the utility of the approach on clustering, classification, etc

Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings

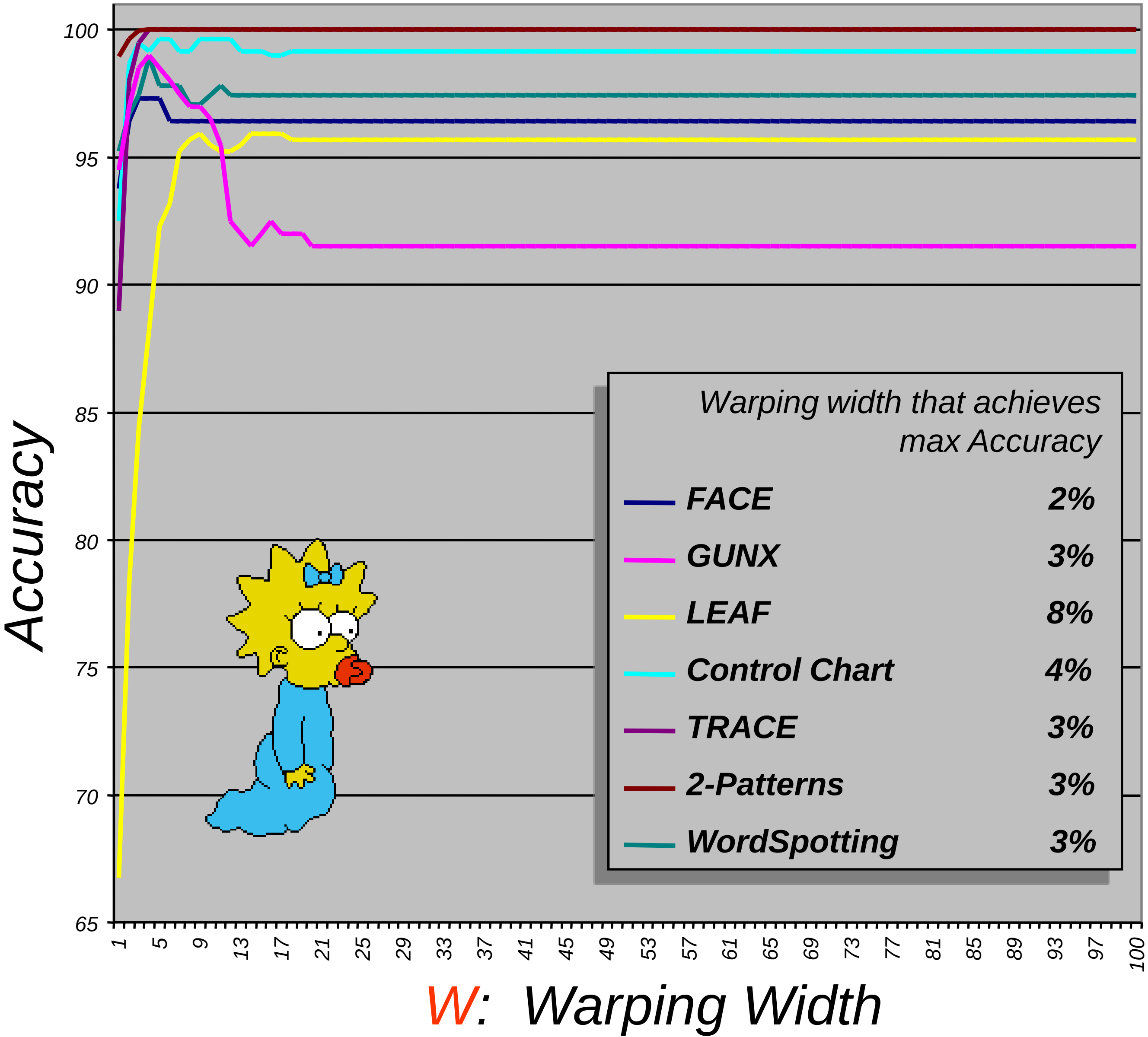


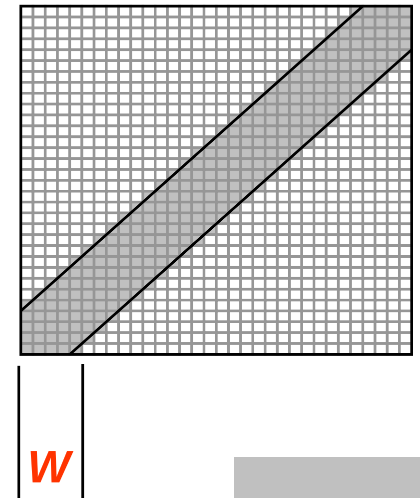
Sakoe-Chiba Band

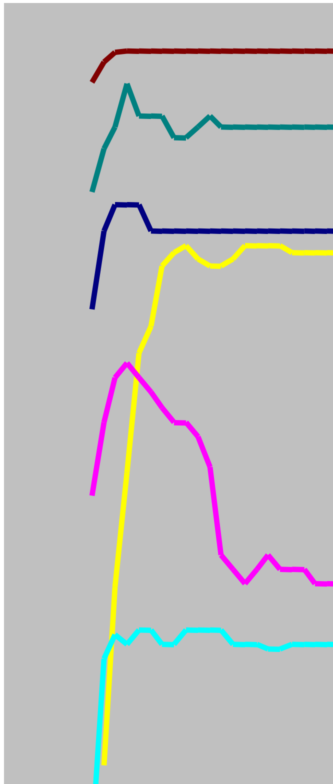


Itakura Parallelogram

Accuracy vs. Width of Warping Window

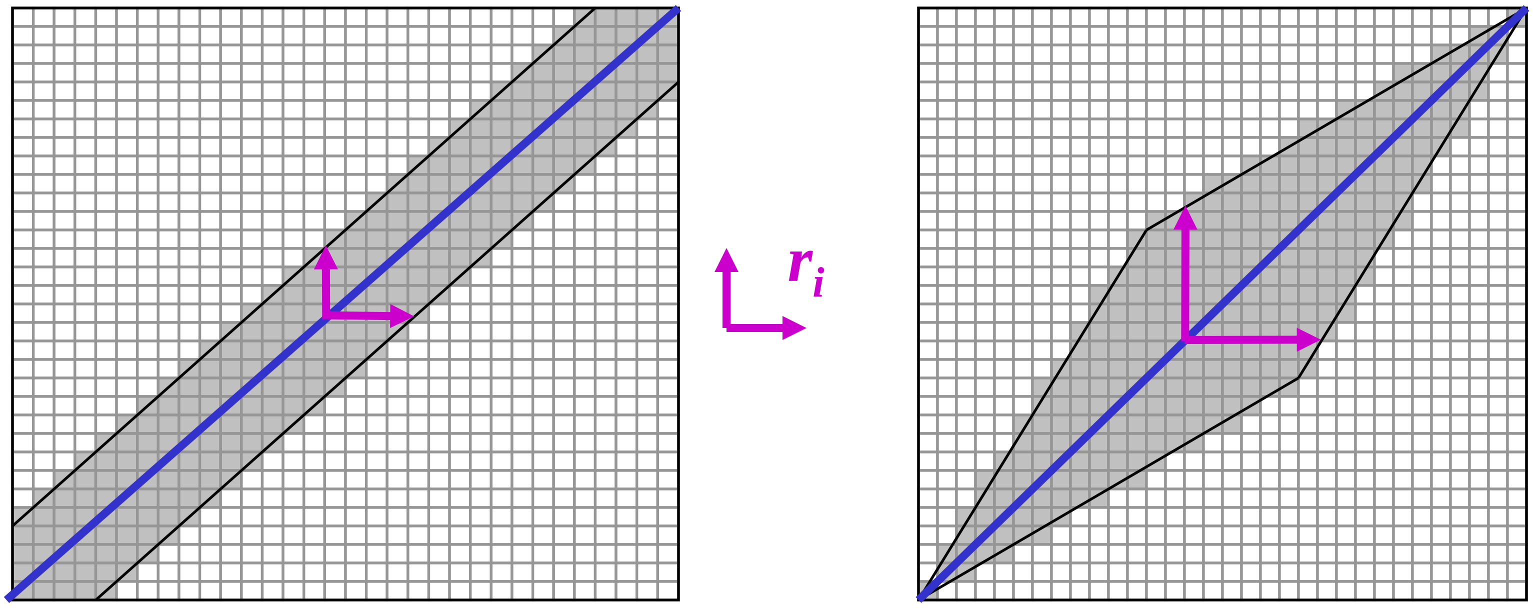






A global constraint constrains the indices of the warping path $w_k = (i,j)_k$ such that $j-r \leq i \leq j+r$

Where r is a term defining allowed range of warping for a given point in a sequence.



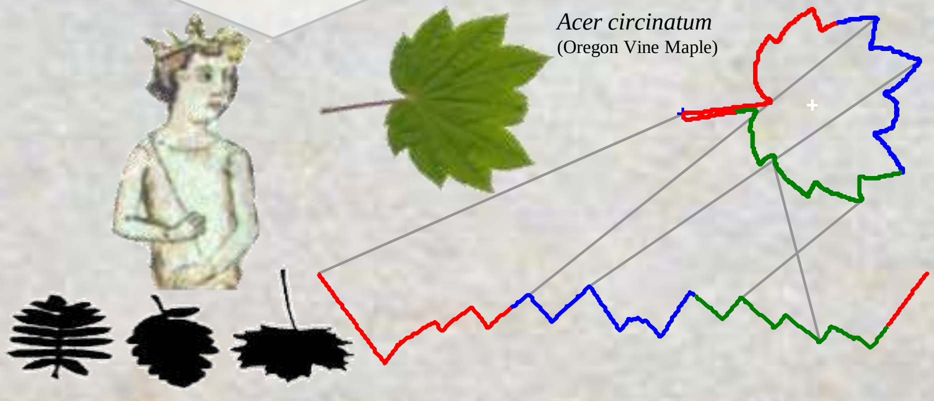
Sakoe-Chiba Band

Itakura Parallelogram

Tests on many diverse datasets

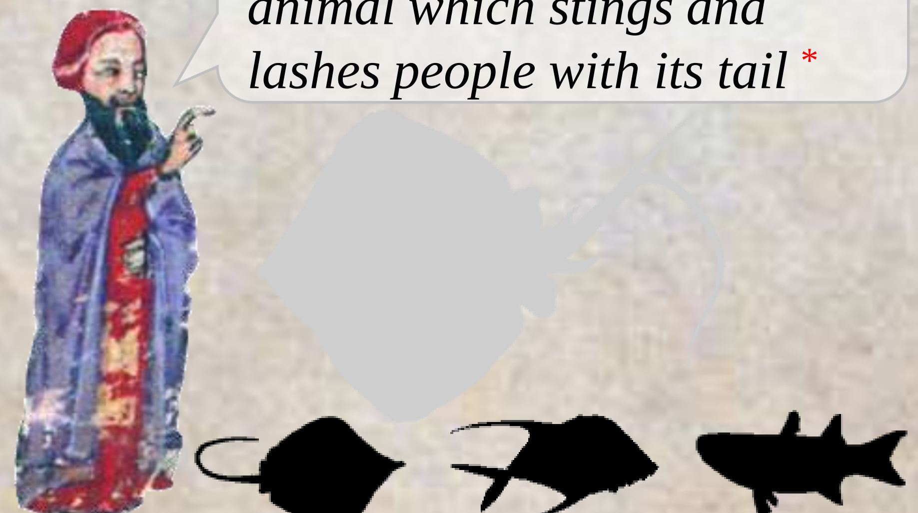
...and I recognized the face ¥

Leaf of mine, in whom I found pleasure ĩ

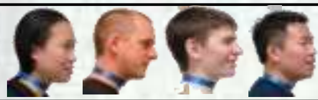
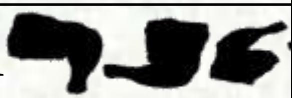


...as a fish dives through water £

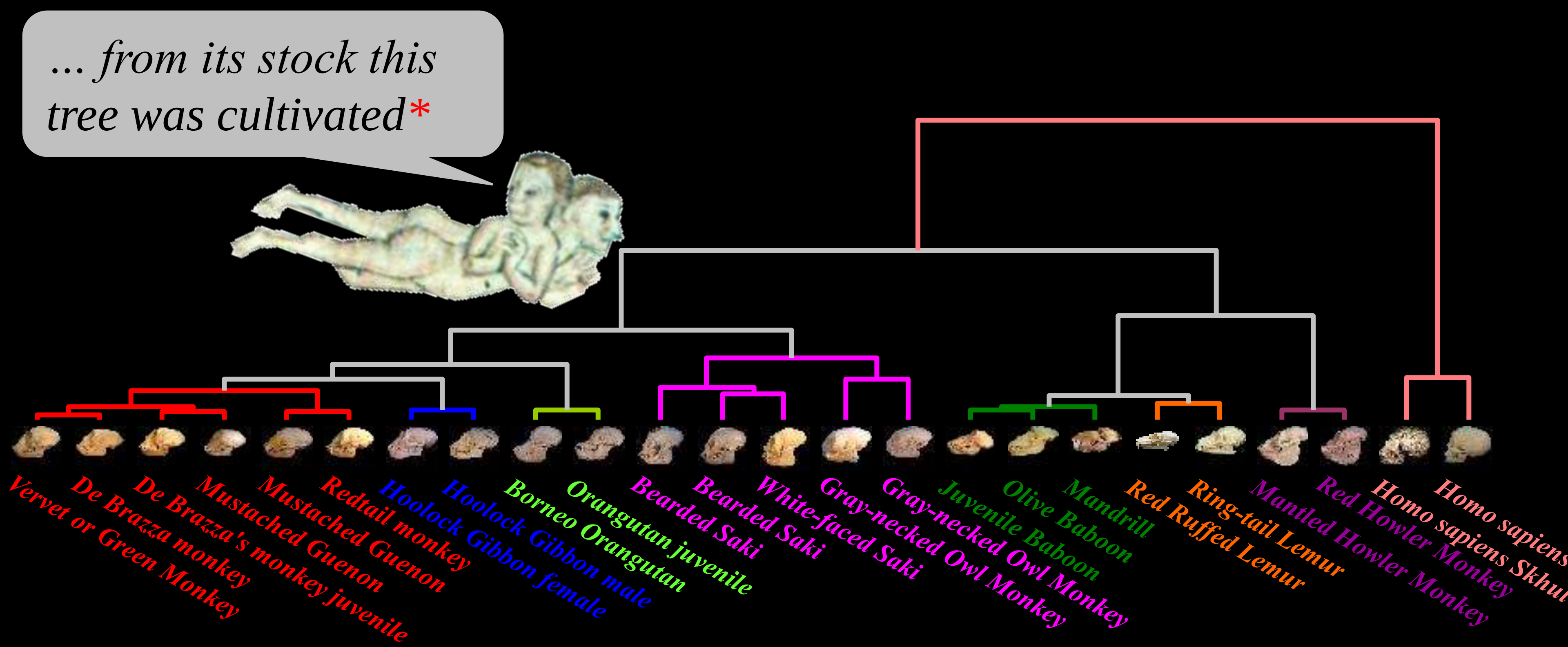
...the shape of that cold animal which stings and lashes people with its tail *



*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*

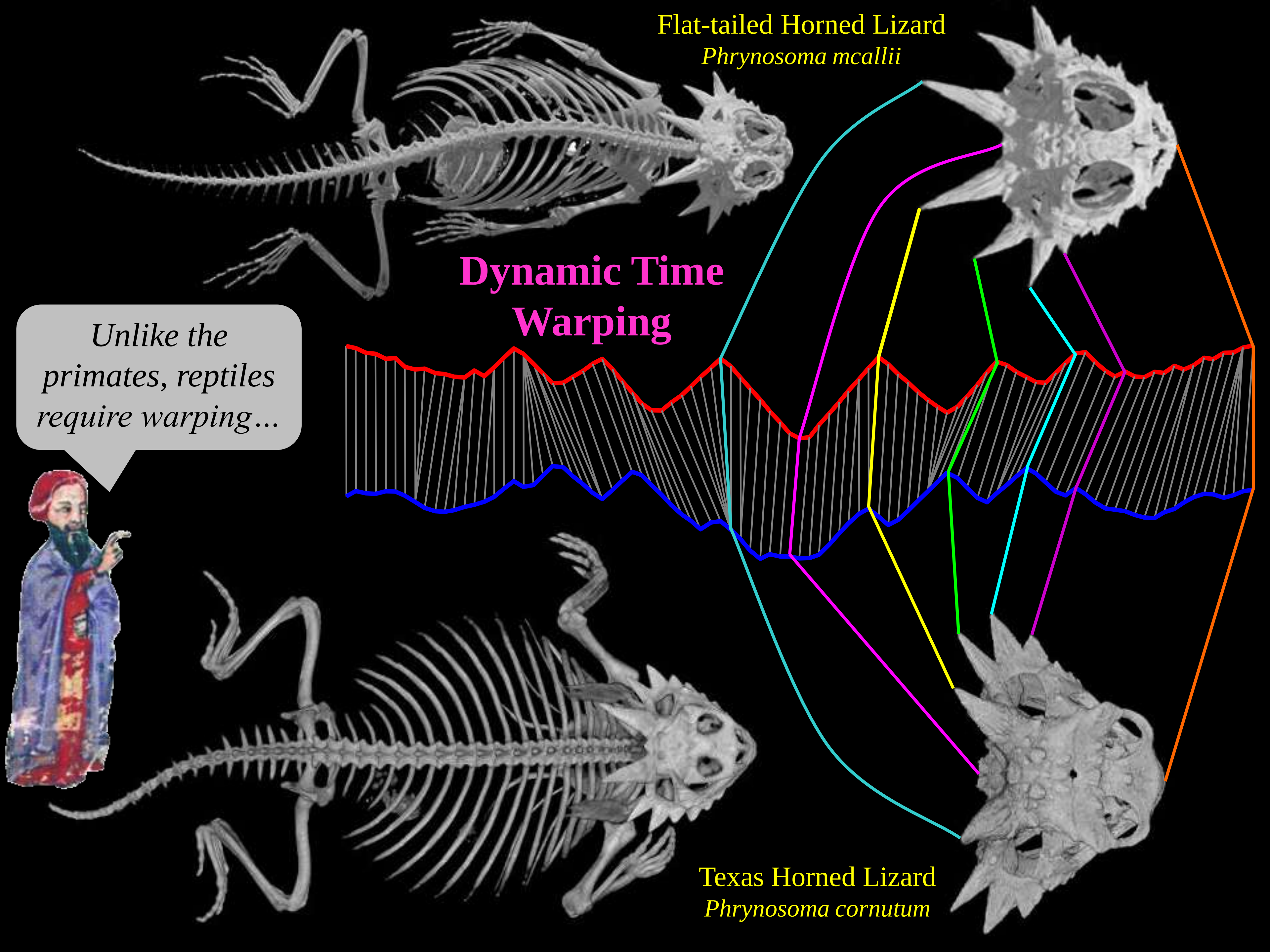
Genus Papio – baboons
Genus Mandrillus- Mandrill

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

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

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

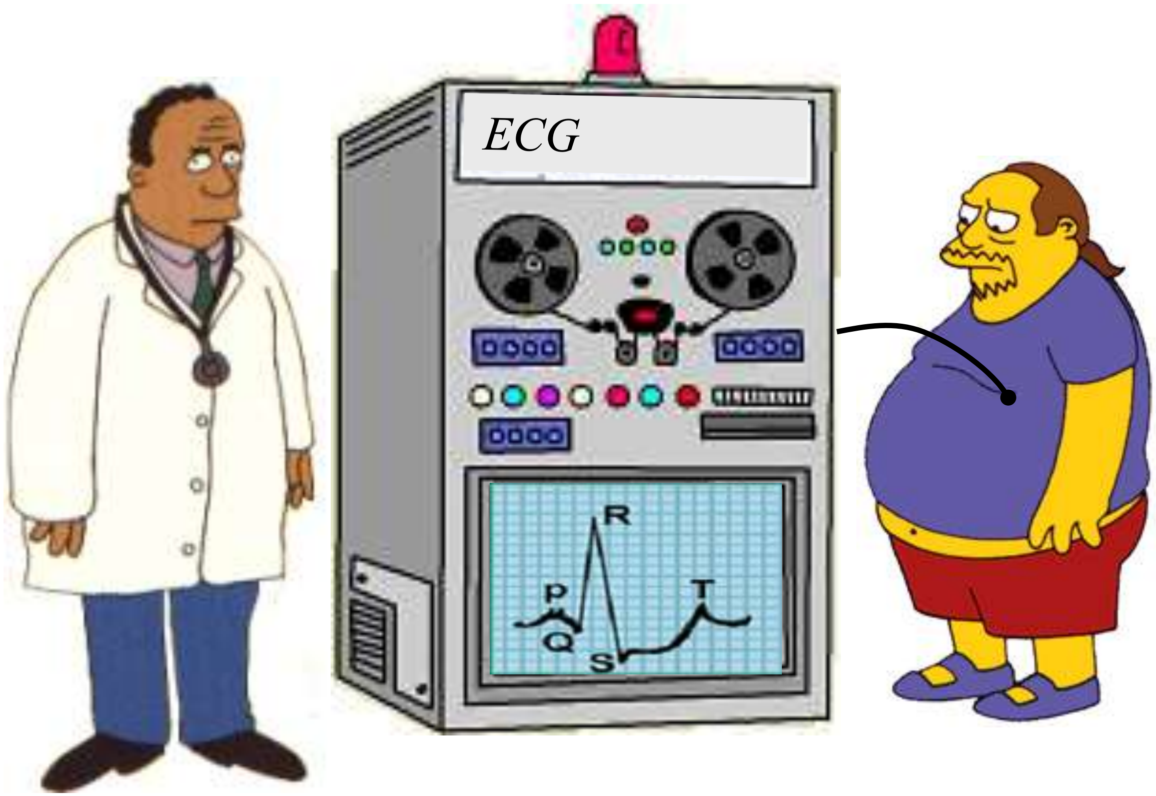
***Purgatorio -- Canto XXIV 117**


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.*

Motivating example revisited…



You go to the doctor because of chest pains. Your ECG looks strange…

Your doctor wants to search a database to find ***similar*** *ECGs, in the hope that they will offer clues about your condition...*

Two questions:

- *How do we define similar?*
- ***How do we search quickly?***

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 $\frac{1}{4}$ gig of data, 100 sec
Clustering $\frac{1}{2}$ gig of data, 200 sec
Clustering 1 gig of data, 400 sec
Clustering 1.1 gigs of data, 20 hours

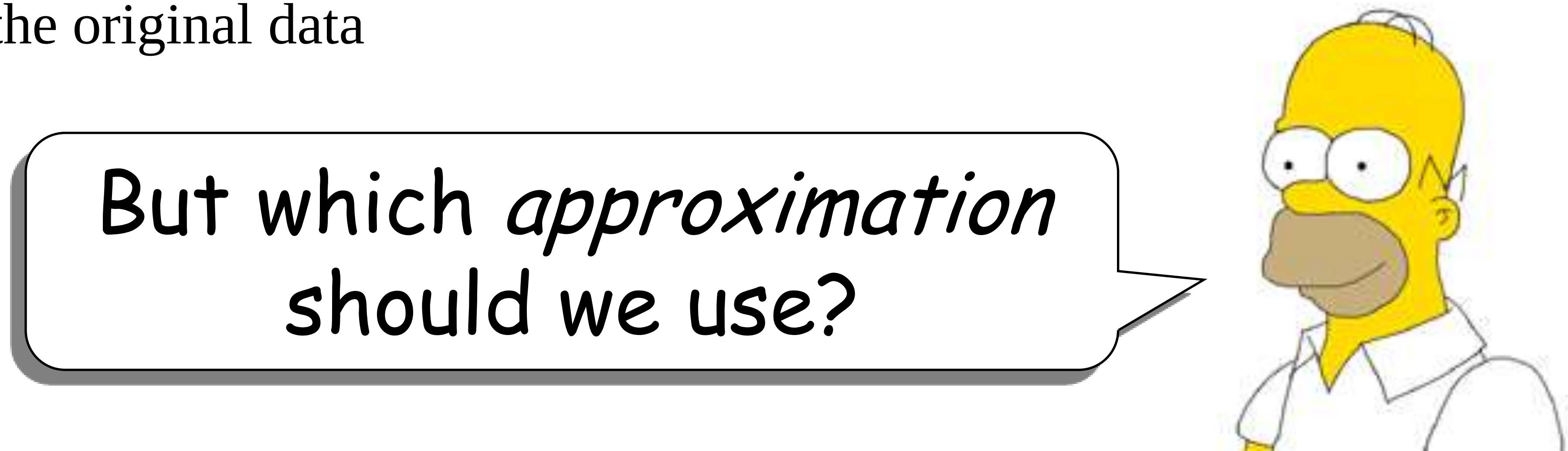
Bradley, M. Fayyad, & Reina: Scaling Clustering Algorithms to Large Databases. KDD 1998: 9-15

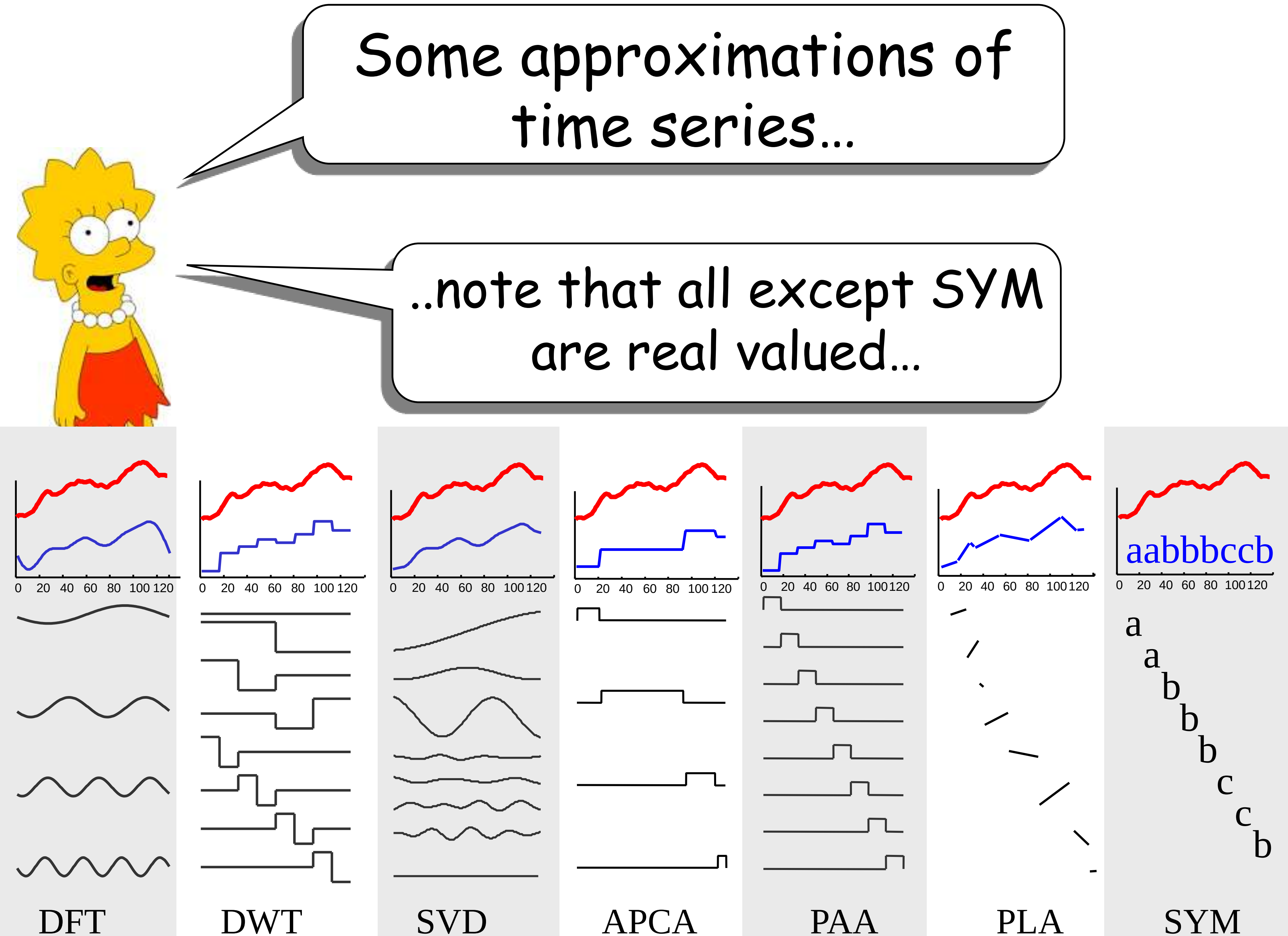
The Generic Data Mining Algorithm

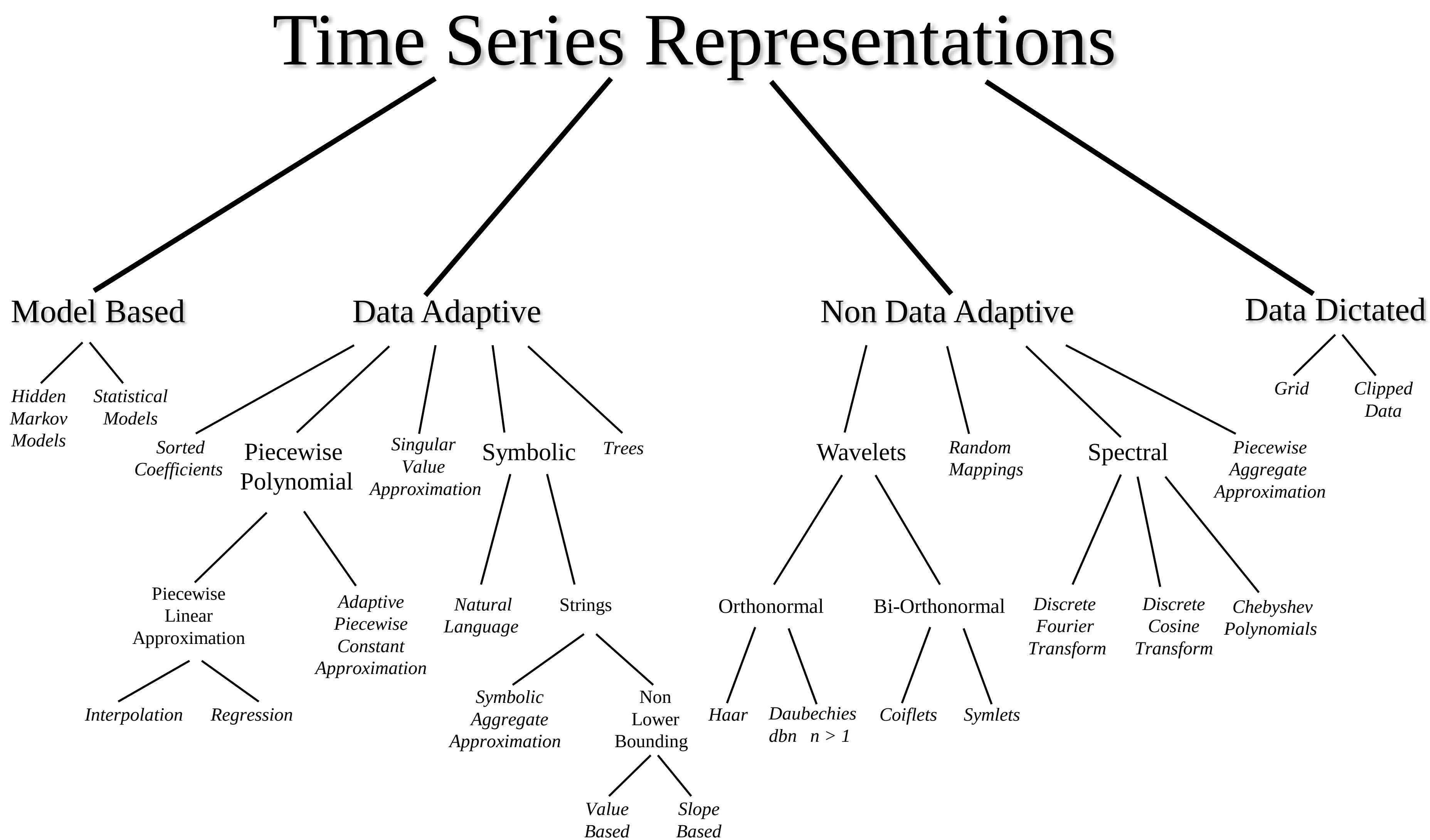
- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest

- 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...
0 20 40 60 80 100 120
aabbbccb
a
a
b
b
b
c
c
b
DFT
DWT
SVD
APCA
PAA
PLA
SYM


Time Series Representations
Model Based
Data Adaptive
Non Data Adaptive
Data Dictated
Hidden Markov Models
Statistical Models
Sorted Coefficients
Piecewise Polynomial
Singular Value Approximation
Symbolic
Trees
Wavelets
Random Mappings
Spectral
Piecewise Aggregate Approximation
Grid
Clipped Data
Piecewise Linear Approximation
Adaptive Piecewise Constant Approximation
Natural Language
Strings
Orthonormal
Bi-Orthonormal
Discrete Fourier Transform
Discrete Cosine Transform
Chebyshev Polynomials
Interpolation
Regression
Symbolic Aggregate Approximation
Non Lower Bounding
Haar
Daubechies dbn n > 1
Coiflets
Symlets
Value Based
Slope Based

The Generic Data Mining Algorithm (revisited)

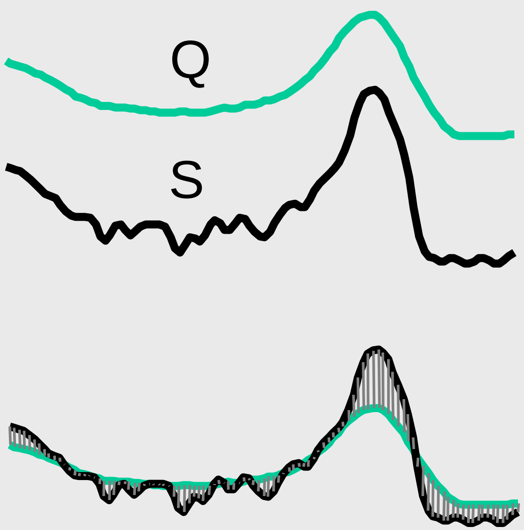
- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest

- 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

What is Lower Bounding?

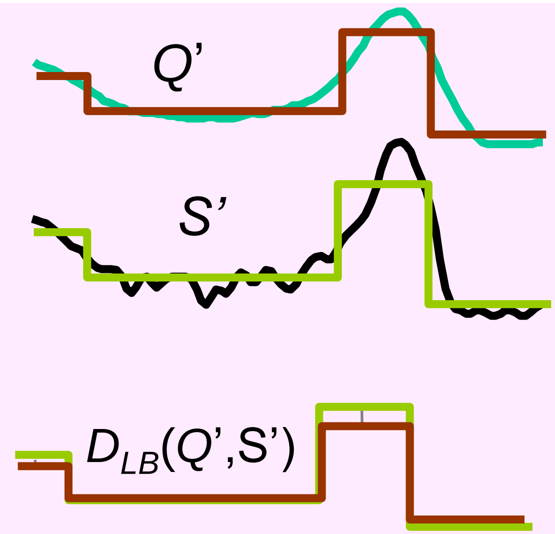
• Lower bounding means the estimated distance in the reduced space is always less than or equal to the distance in the original space.



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

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

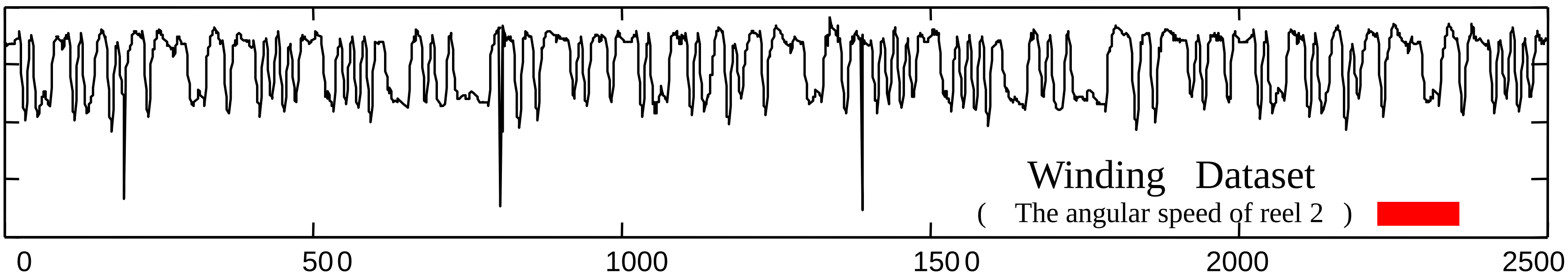
While there are more than 200 different symbolic or discrete ways to approximate time series, none except SAX allows lower bounding

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

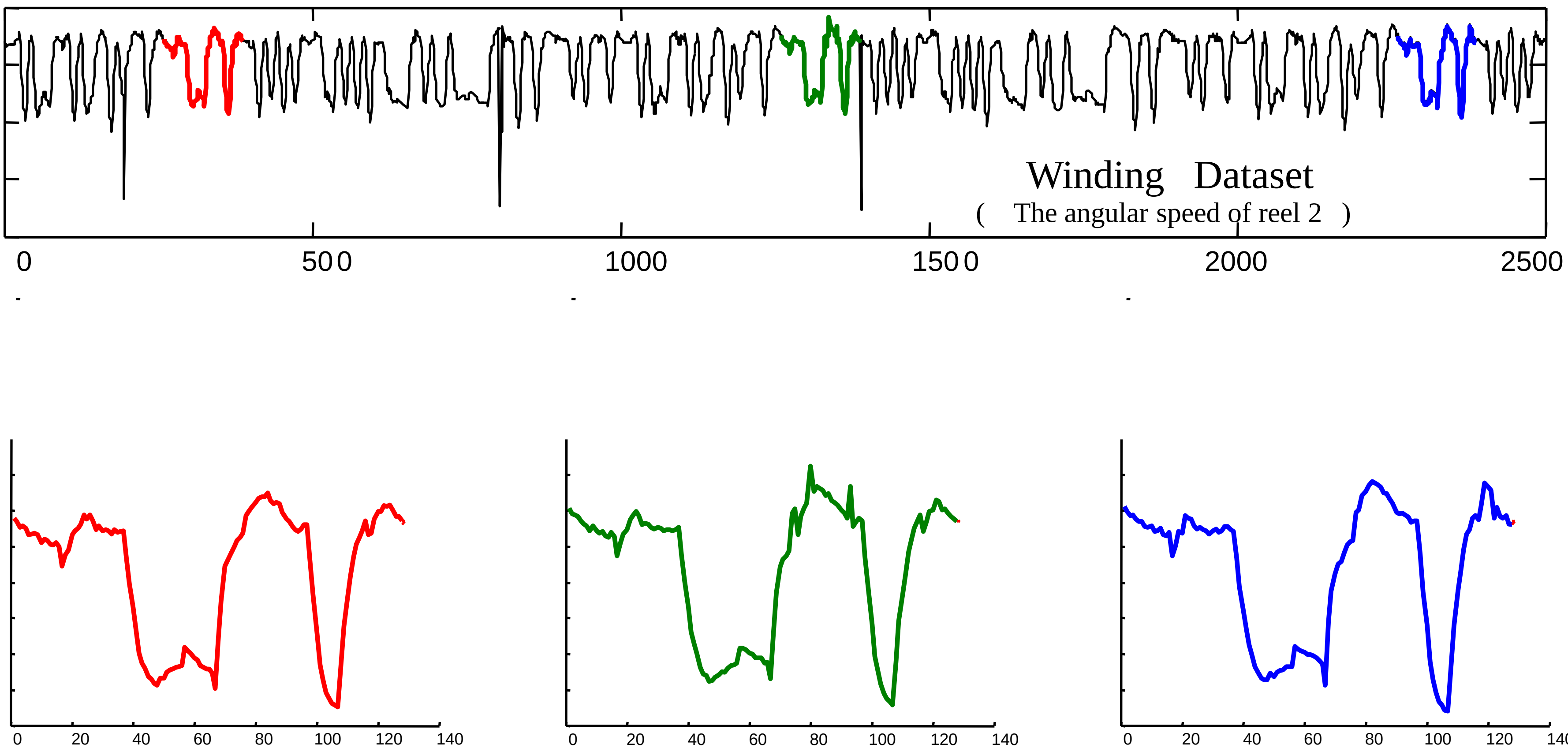
Time Series Motif Discovery

(finding repeated patterns)

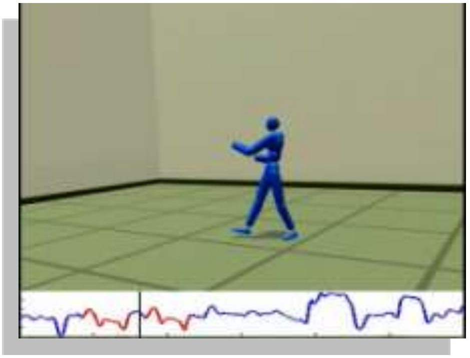


Time Series Motif Discovery

(finding repeated patterns)



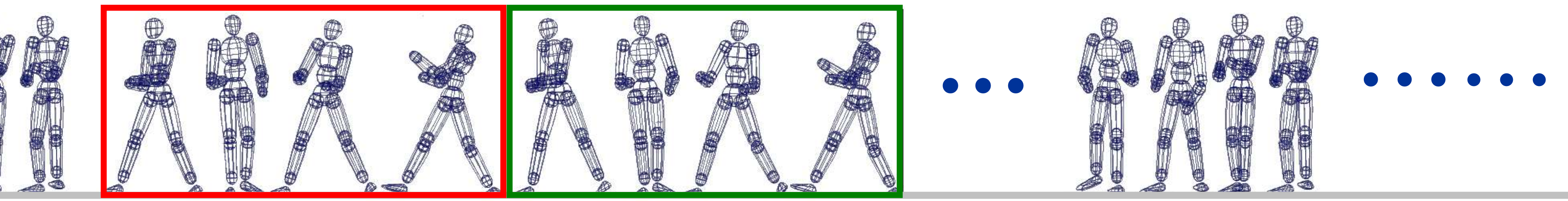
Why Find Motifs? I

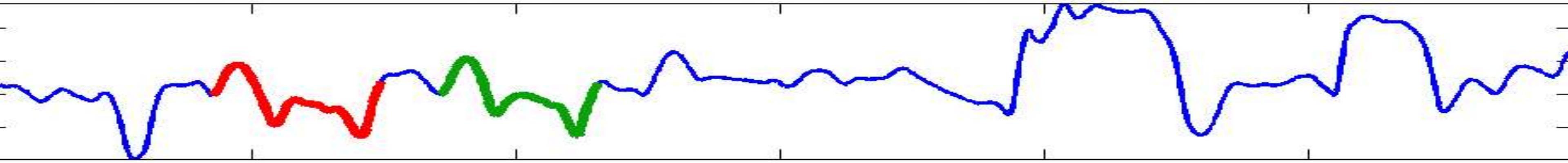


To see the full video go to..
www.cs.ucr.edu/~eamonn/SIGKDD07/UniformScaling.html
Or search YouTube for “Time series motifs ”

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

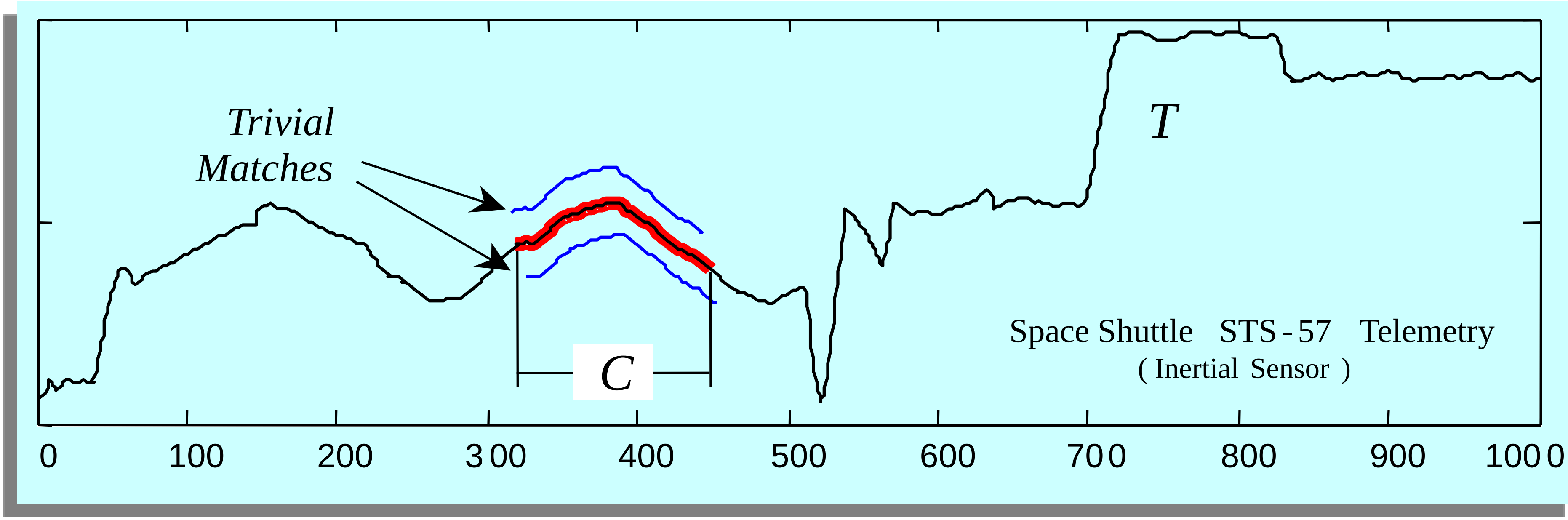
· Mining **association rules** in time series requires the discovery of motifs. These are referred to as *primitive shapes* and *frequent patterns*.

· 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 $1 \leq i < K$.

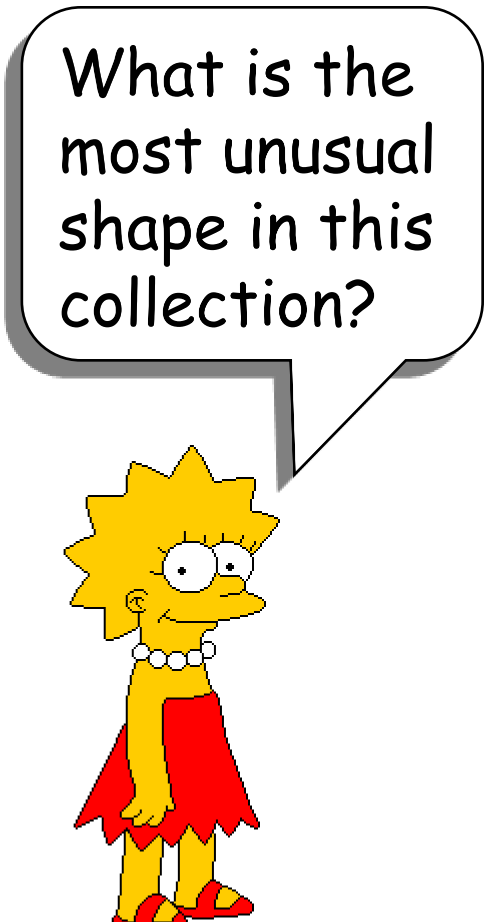
OK, we can define motifs, but how do we find them?

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.**

Image Discords



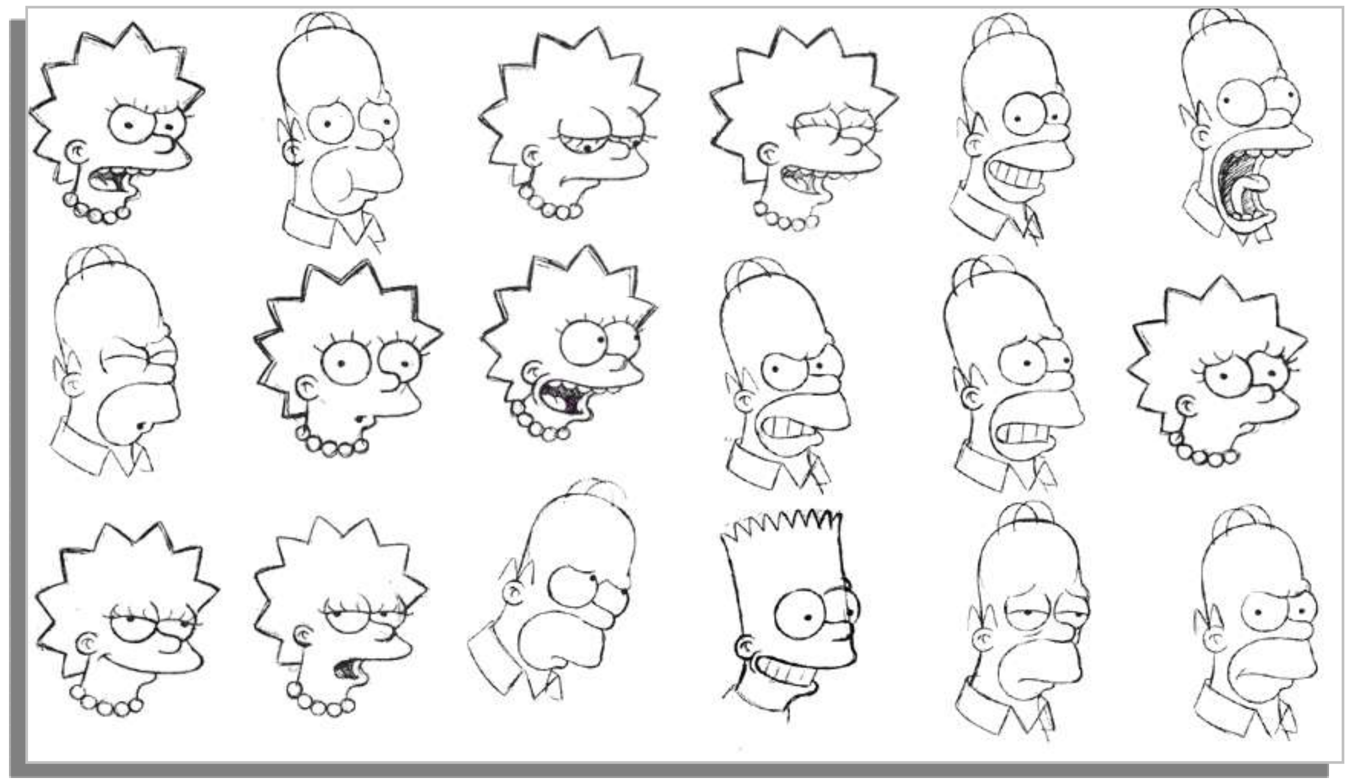
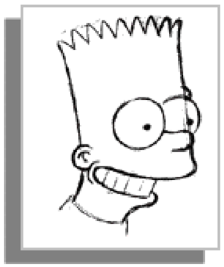
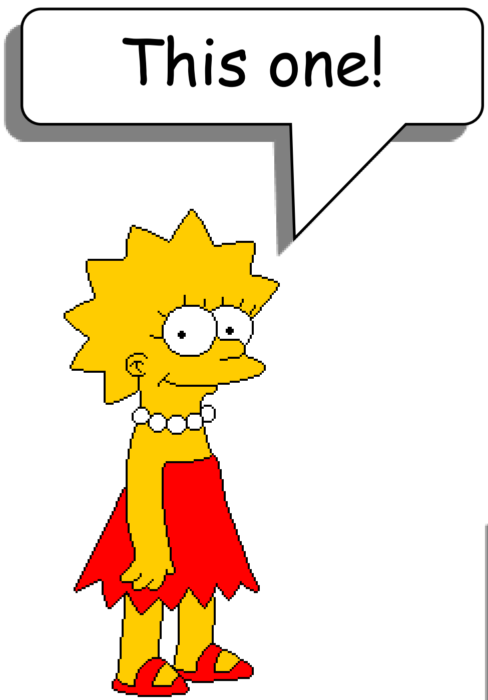
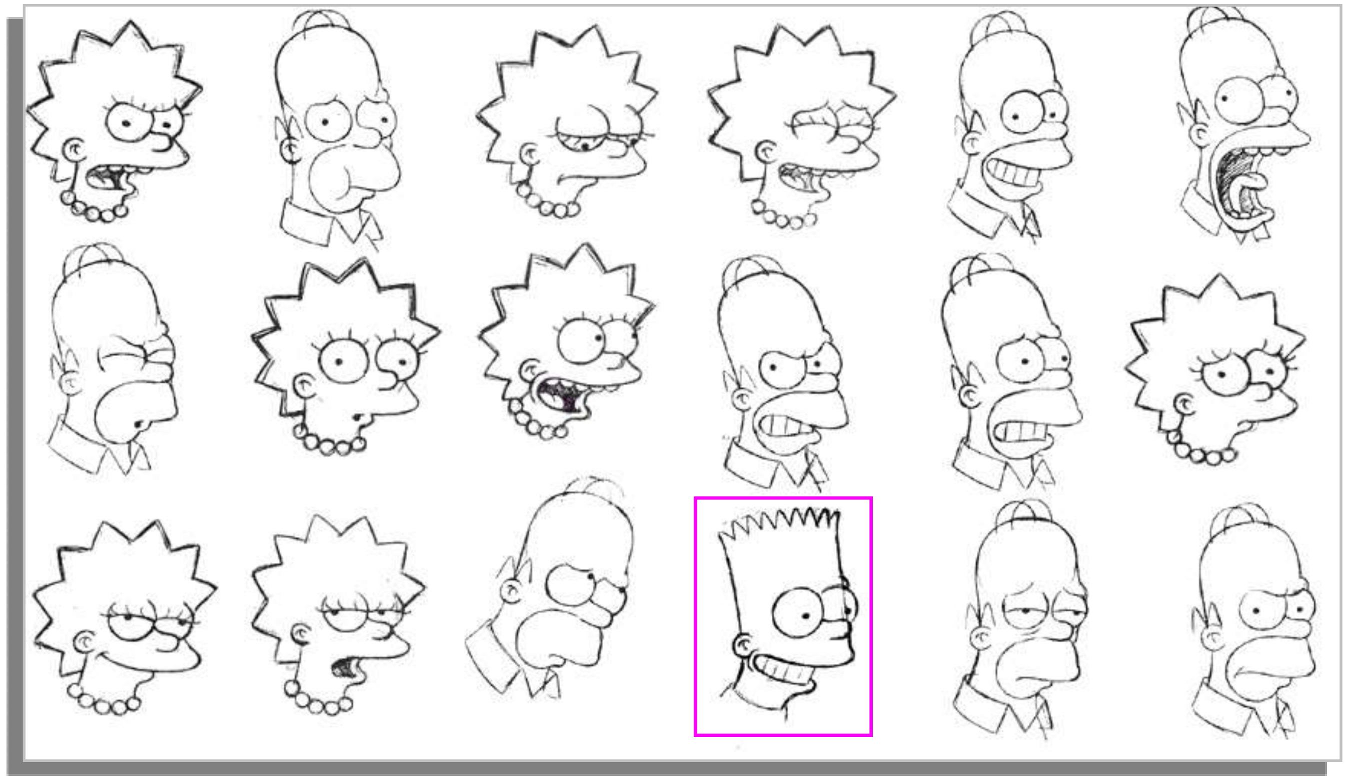


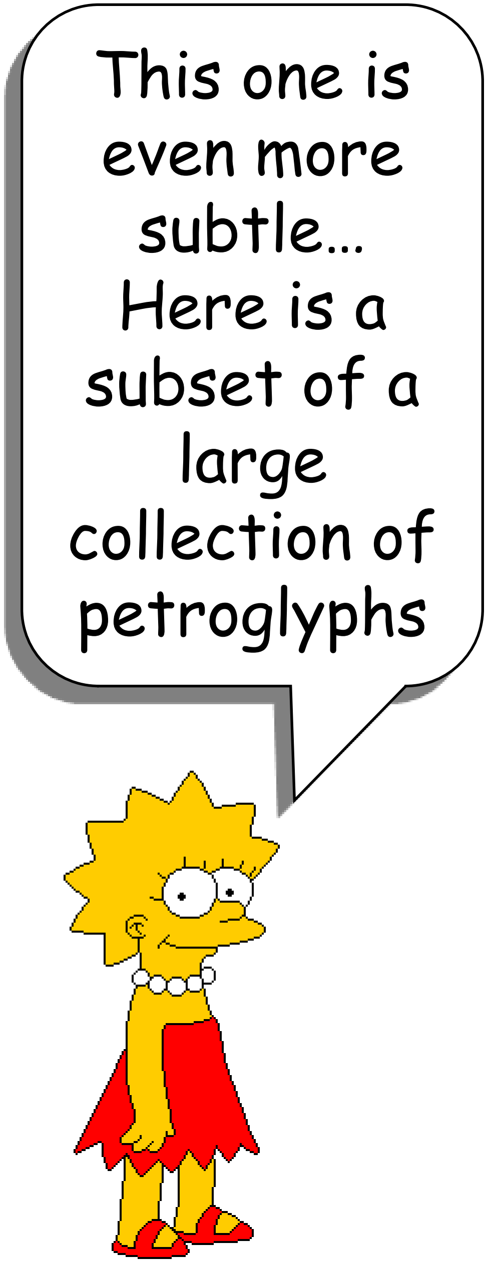
Image Discords

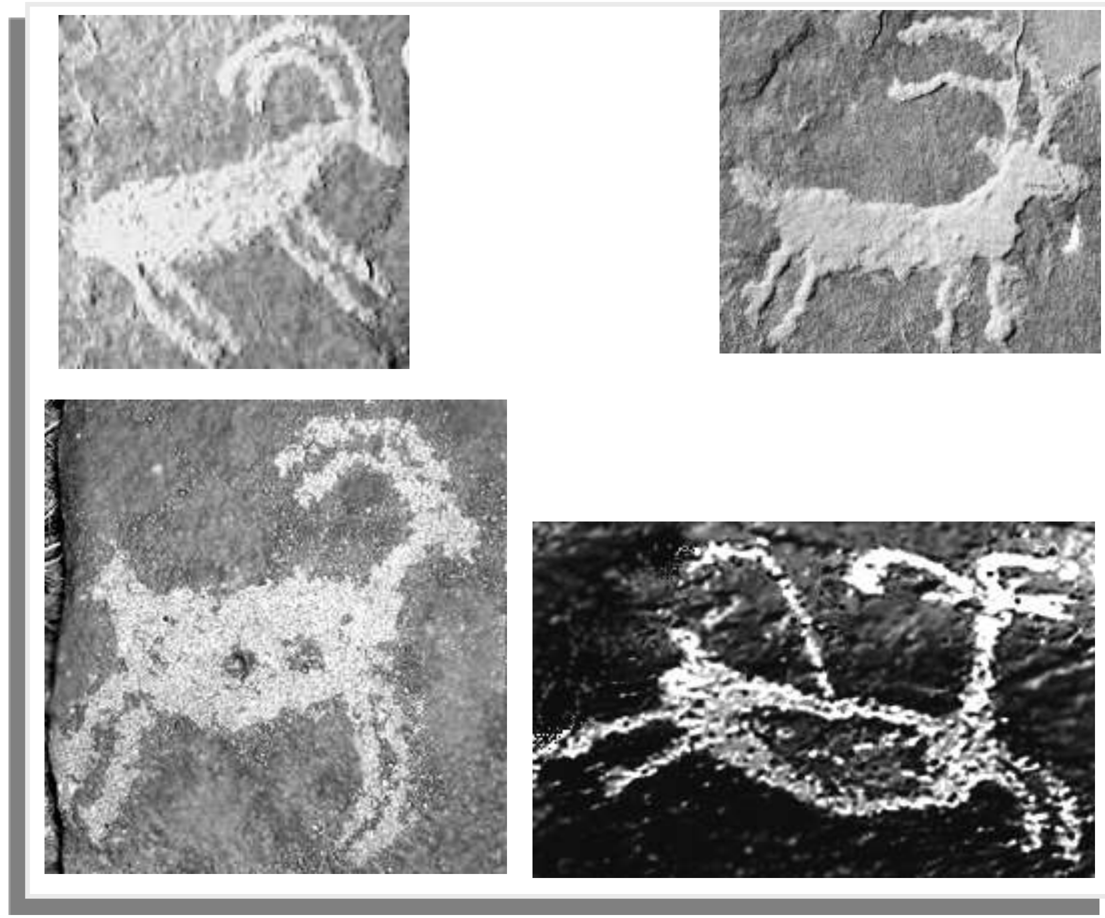


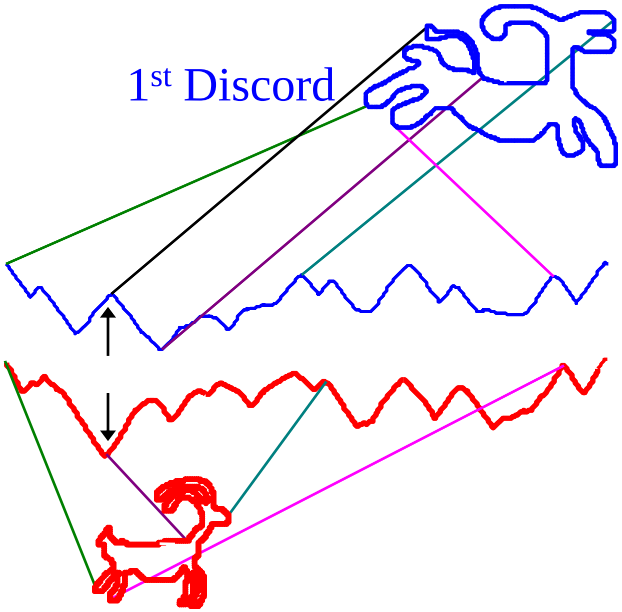


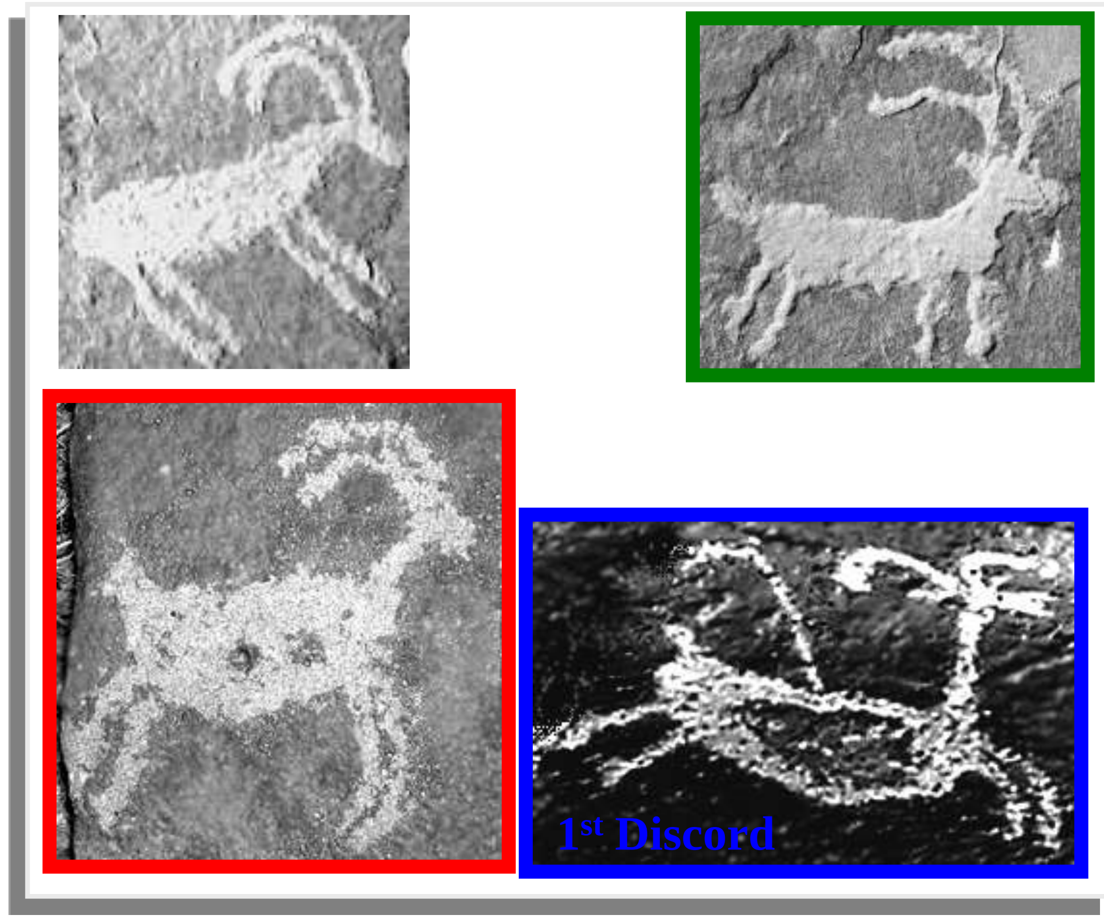


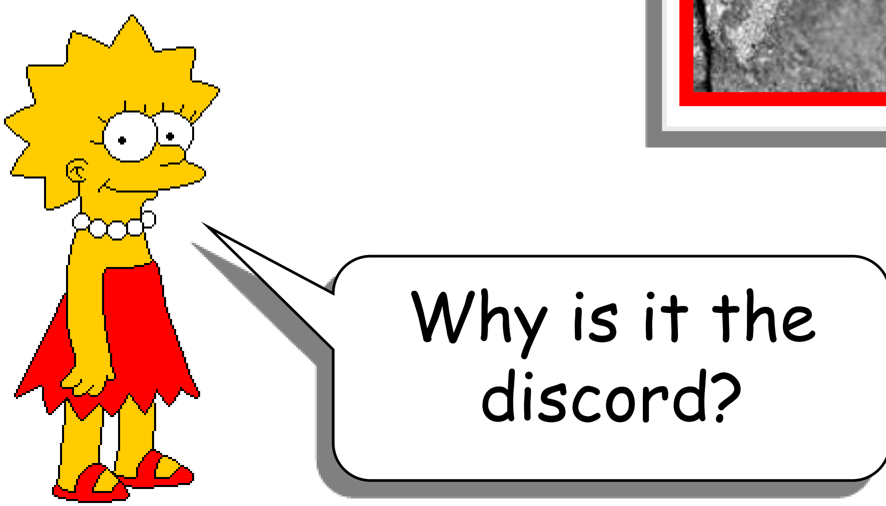
Shape Discord: Given a collection of shapes S, the shape D is the discord of S if D has the largest distance to its nearest match. That is, $\forall$ shapes C in S, the nearest match M_C of C and the nearest match M_D of D, $Dist(D, M_D) > Dist(C, M_C)$.


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




1st Discord


1st Discord


Why is it the discord?

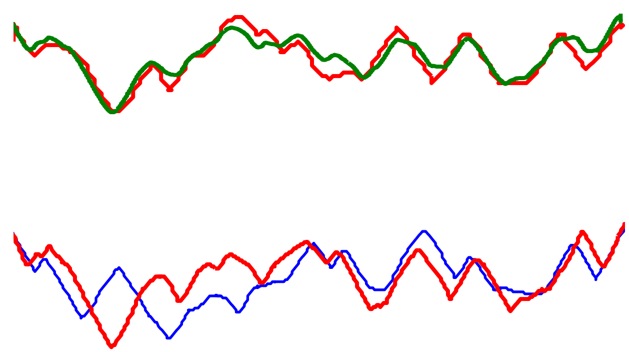
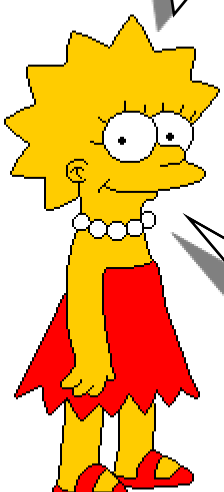
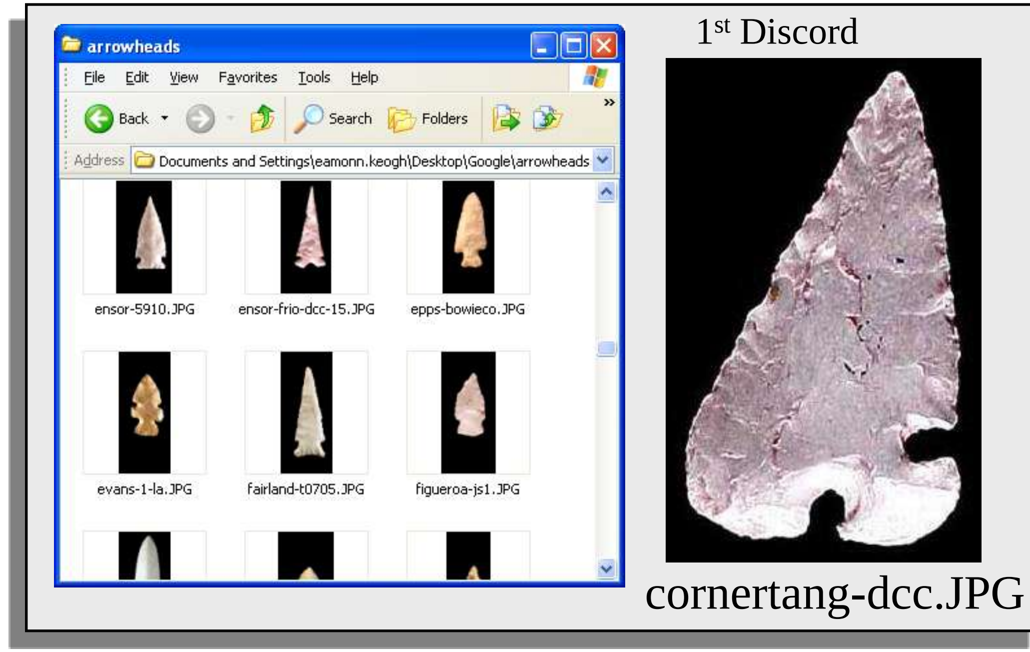
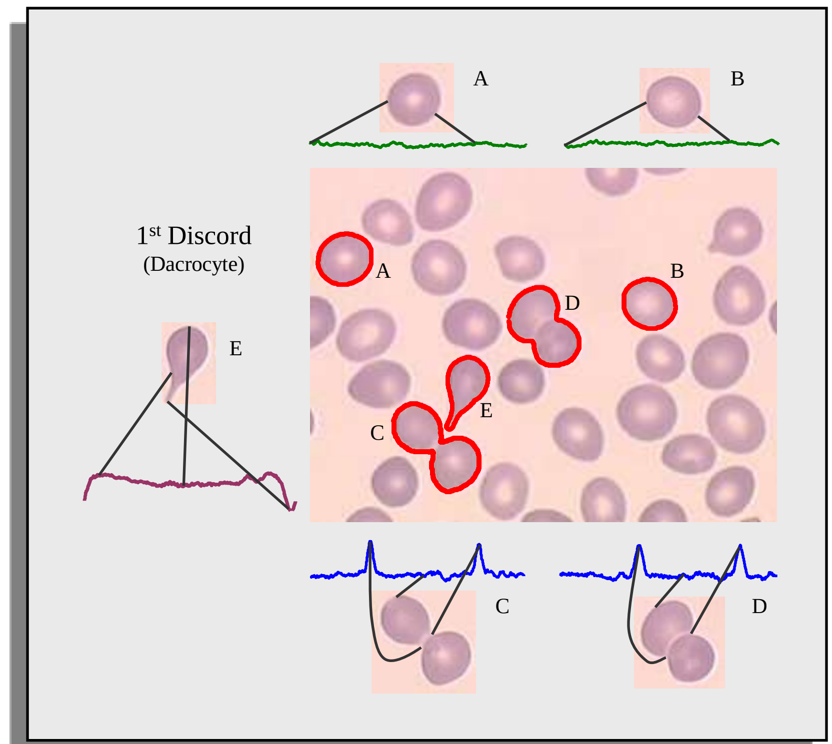


Image discords are potentially useful in many domains…
Most arrowheads are symmetric, but…



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 ]
```

The code says…
Find the smallest (non diagonal) value in each column, the largest of these is the discord

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 ]
```

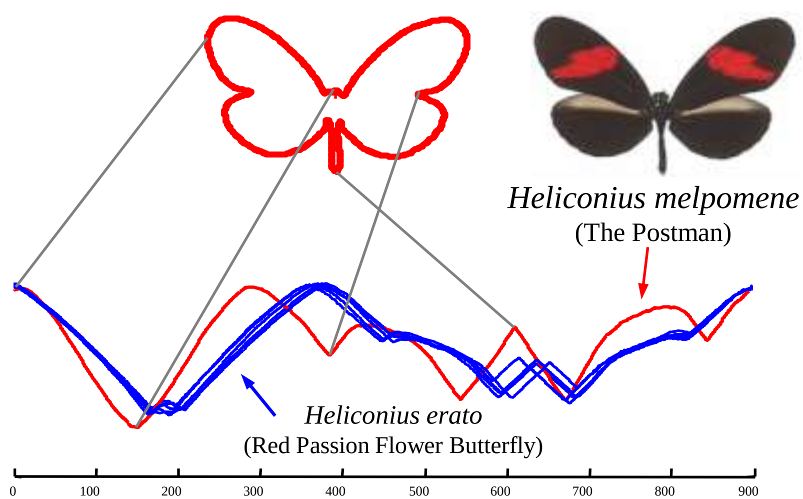
0	2	4.2	1.1	2.3	8.5
2	0	3	3.2	3.5	8.2
4.2	3	0	1.2	9.2	9.7
1.1	3.2	1.2	0	0.1	7.5
2.3	3.5	9.2	0.1	0	7.6
8.5	8.8	9.7	7.5	7.6	0

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.

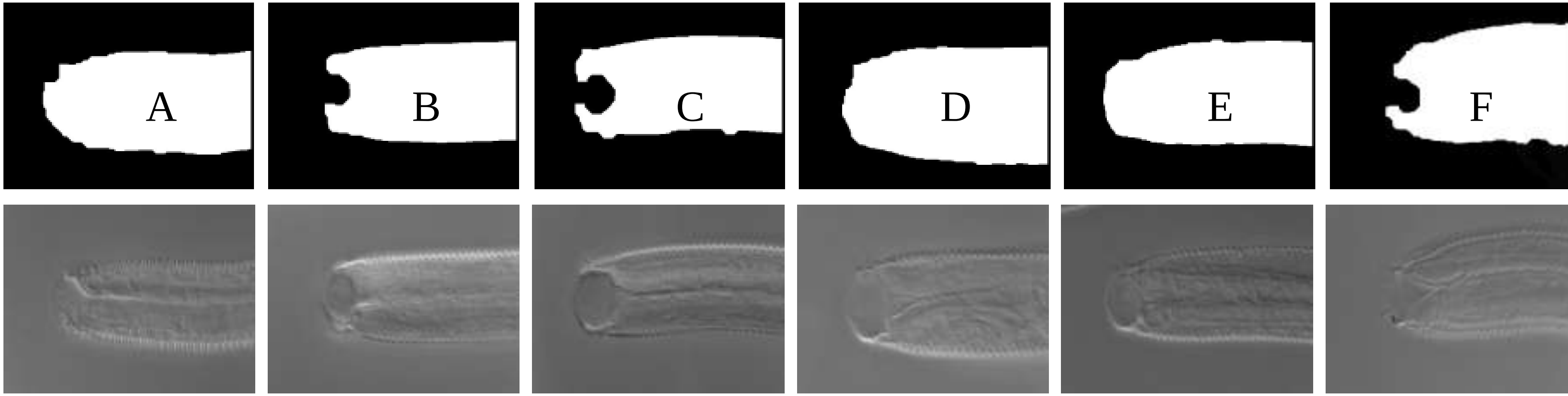
The code also uses heuristics to order the search…

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

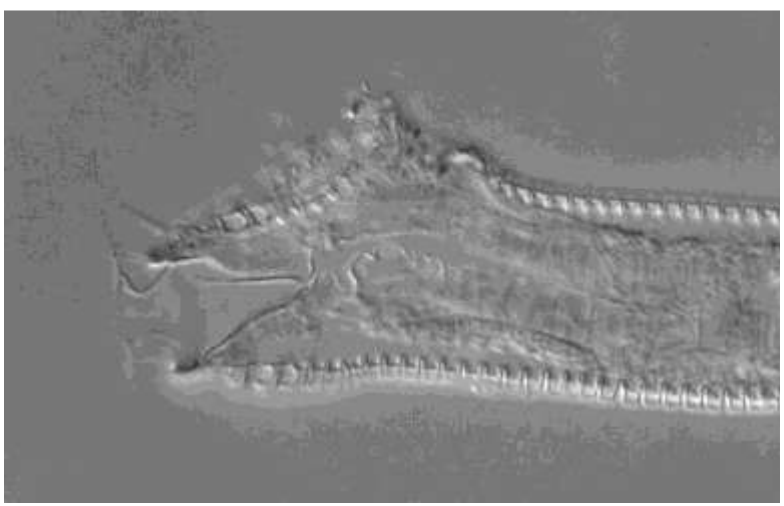
One of them is *not* a Red Passion Flower Butterfly. A fact that can be discovered by finding the shape discord

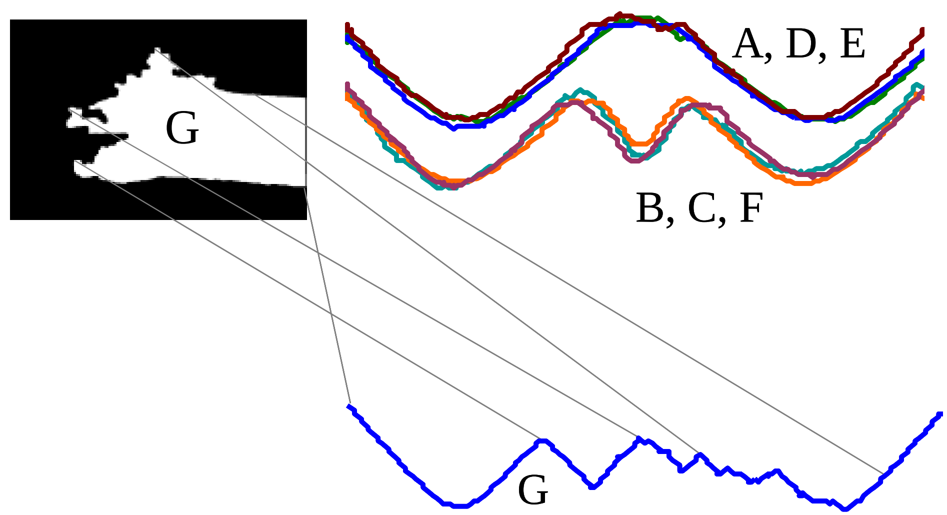


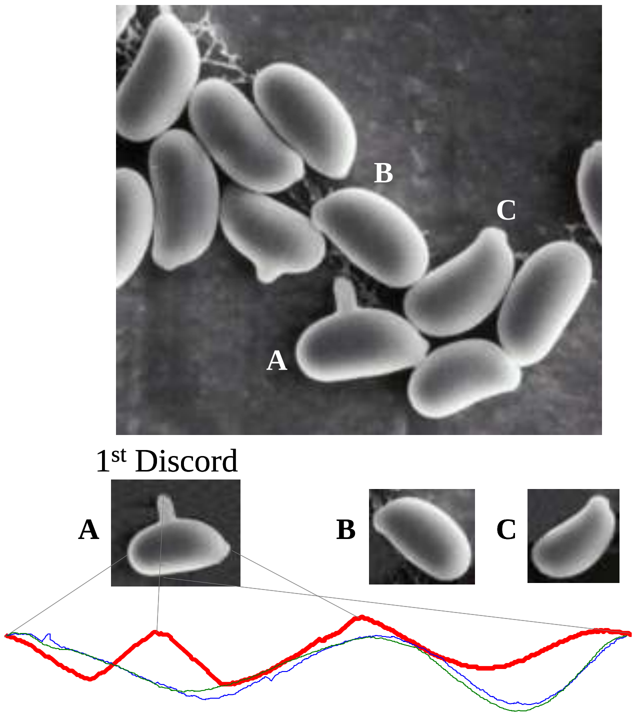
Nematode Discords



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

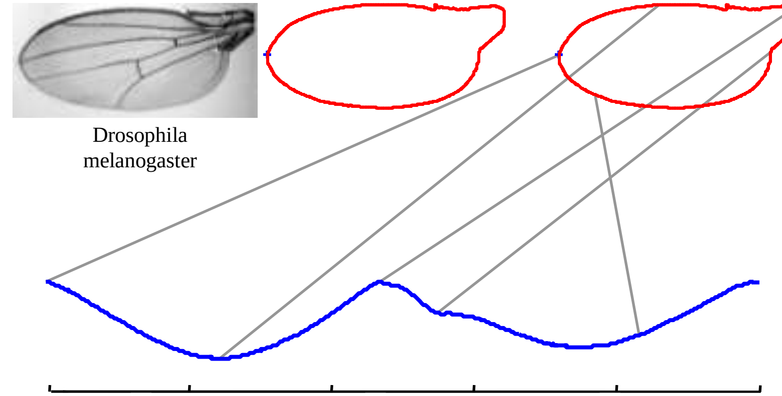


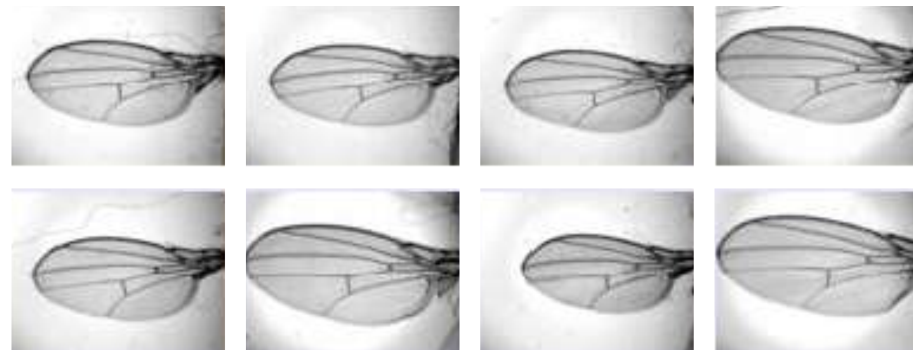




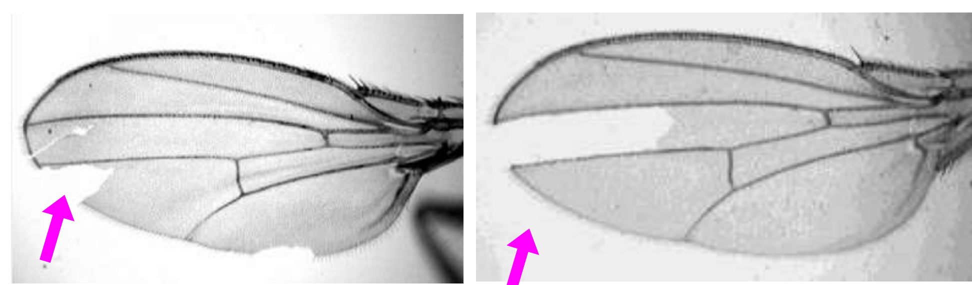
Fungus Images

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.



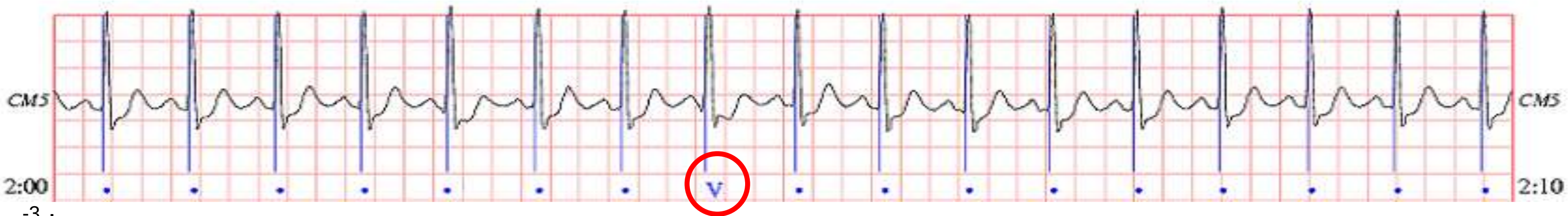


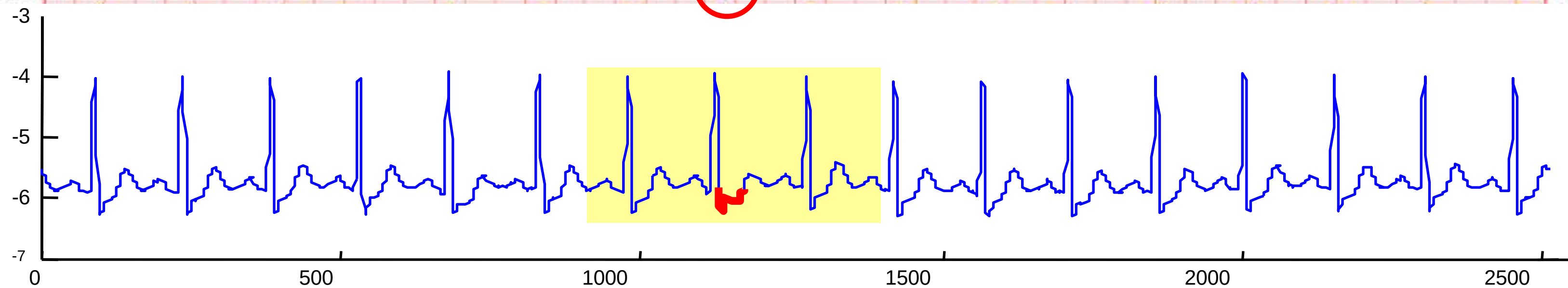
A subset of 32,028 images of Drosophila wings



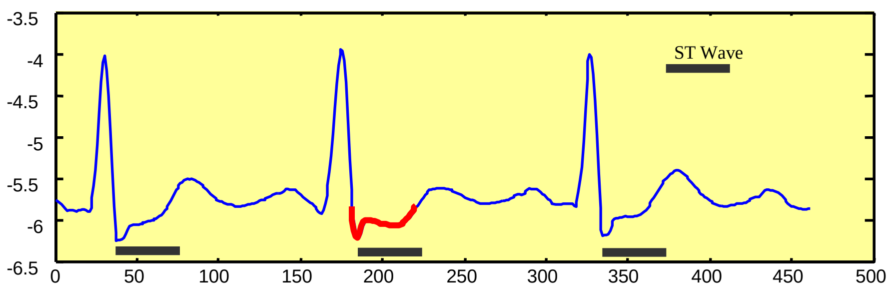
Discords in Medical Data

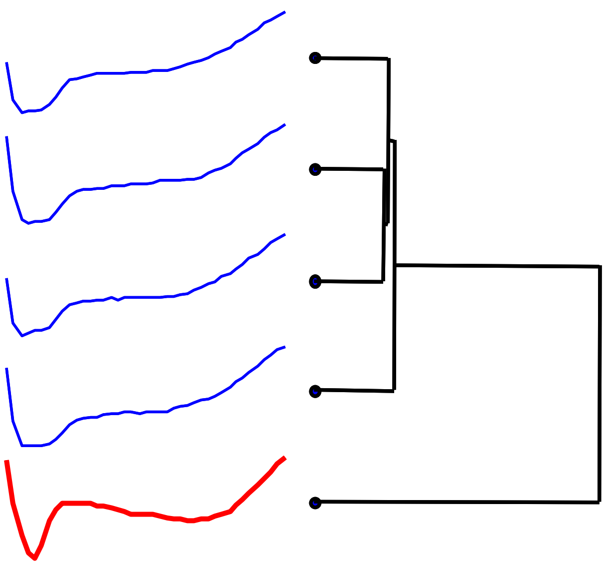
A cardiologist noted subtle anomalies in this dataset. Let us see if the discord algorithm can find them.



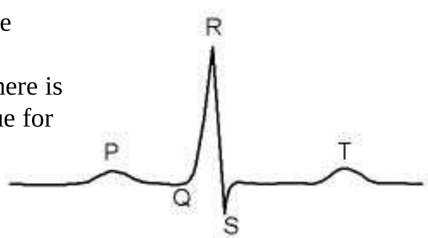


Record qtdbsele0606 from the PhysioBank QT Database (qtdb)





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



And Now For Our Work

Optimal Subsequence Bijection

Suzan Köknar-Tezel (tezel@temple.edu)
Longin Jan Latecki
Qiang Wang
Vasileios Megalooikonomou
Department of Computer and Information Sciences
Temple University
Philadelphia, Pennsylvania

Outline

- What is OSB?
- Experimental results
  - See appendix for tables and graphs
- Terminology and definitions
- Motivation
- The algorithm
- A simple example
- Calculating the jumpcost

What is OSB?

- We consider the problem of elastic matching of sequences of real numbers
- When matching, it is desirable to exclude the outlier elements in order to obtain a robust matching performance
- In many applications it is also desirable to have a bijection between the remaining elements
- OSB is an algorithm that determines the optimal subsequence bijection between two sequences of real numbers

Experimental Results

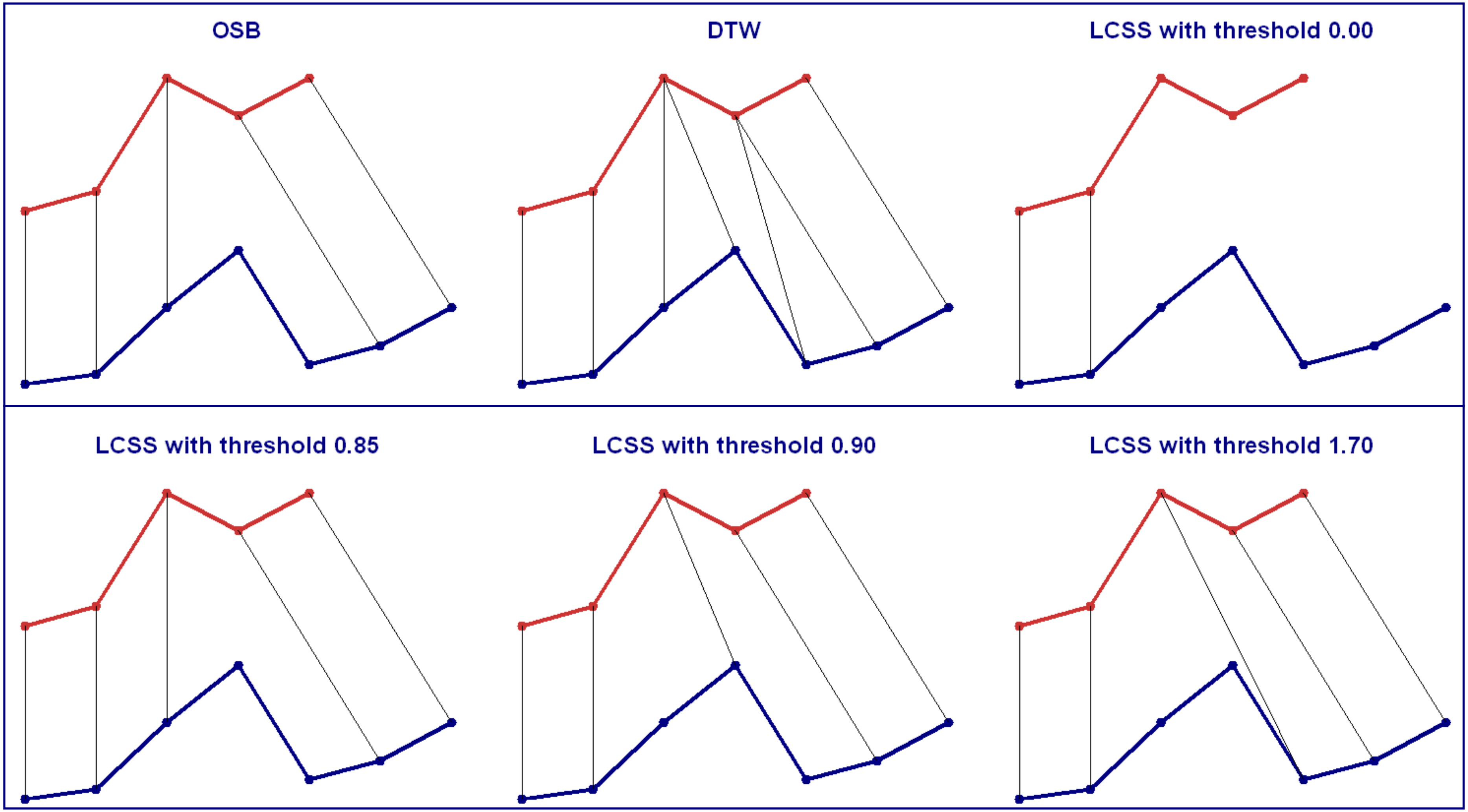
- We tested our method on 3 groups of data
  - The KDD 2007 competition datasets (20 datasets)
    - We were first on 3 datasets and second on 1 dataset
  - The UCR datasets (20 datasets)
    - We had best accuracy on 10 datasets
    - We tied for best on 3 datasets
  - The MPEG 7 dataset (partial shape matching)
    - We had 100% recall rate for 1NN and 2NN
    - We had 67% recall rate for 20NN

Terminology and Definitions

- OSB – Optimal Subsequence Bijection
- DTW – Dynamic Time Warping
- LCSS – Longest Common SubSequence
- Sequences:
  - $a = (a_1, \ldots, a_m)$, $b = (b_1, \ldots, b_n)$
- $d(a_i, b_j)$ is the "distance" between element a_i in a and element b_j in b
- C – Jump cost – the penalty for skipping an element
- DAG – Directed Acyclic Graph

Motivation

Example sequences:
a = {1, 2, 8, 6, 8}
b = {1, 2, 9, 15, 3, 5, 9}



OSB Algorithm

- Goal: given two real-valued sequences ***a*** and ***b***, find subsequences ***a′*** of ***a*** and ***b′*** of ***b*** such that ***a′*** best matches ***b′***
  - Possible to skip elements in both ***a*** and ***b***
    - The ability to exclude outliers
  - Preserve the order of the elements
  - A one-to-one correspondence

OSB Algorithm (2)

- Create a dissimilarity matrix
  - No restrictions on the distance function d
    - We used $d(a_i, b_j) = (a_i - b_j)^2$
- To find the optimal correspondence, use a shortest path algorithm on a DAG

OSB Algorithm (3)

- The nodes of the DAG are all the index pairs of the matrix: (i,j)$\in${1,…,m}$\times${1,…,n}
- The edge weights w are defined by

$$w((i,j),(k,l)) = \begin{cases} \sqrt{(k-i-1)^2 + (l-j-2)^2} \cdot C + d(a_k, b_l) & \text{if } i < k \wedge j < l \\ \infty & \text{otherwise} \end{cases}$$

  - C is the jump cost (the penalty for skipping an element)

OSB Algorithm (4)

- The edge cost may be extended to impose a warping window
  - Set a maximal value for $k - i - 1$ and $l - j - 1$
- This definition of the edge weights is our main contribution

A Simple Example

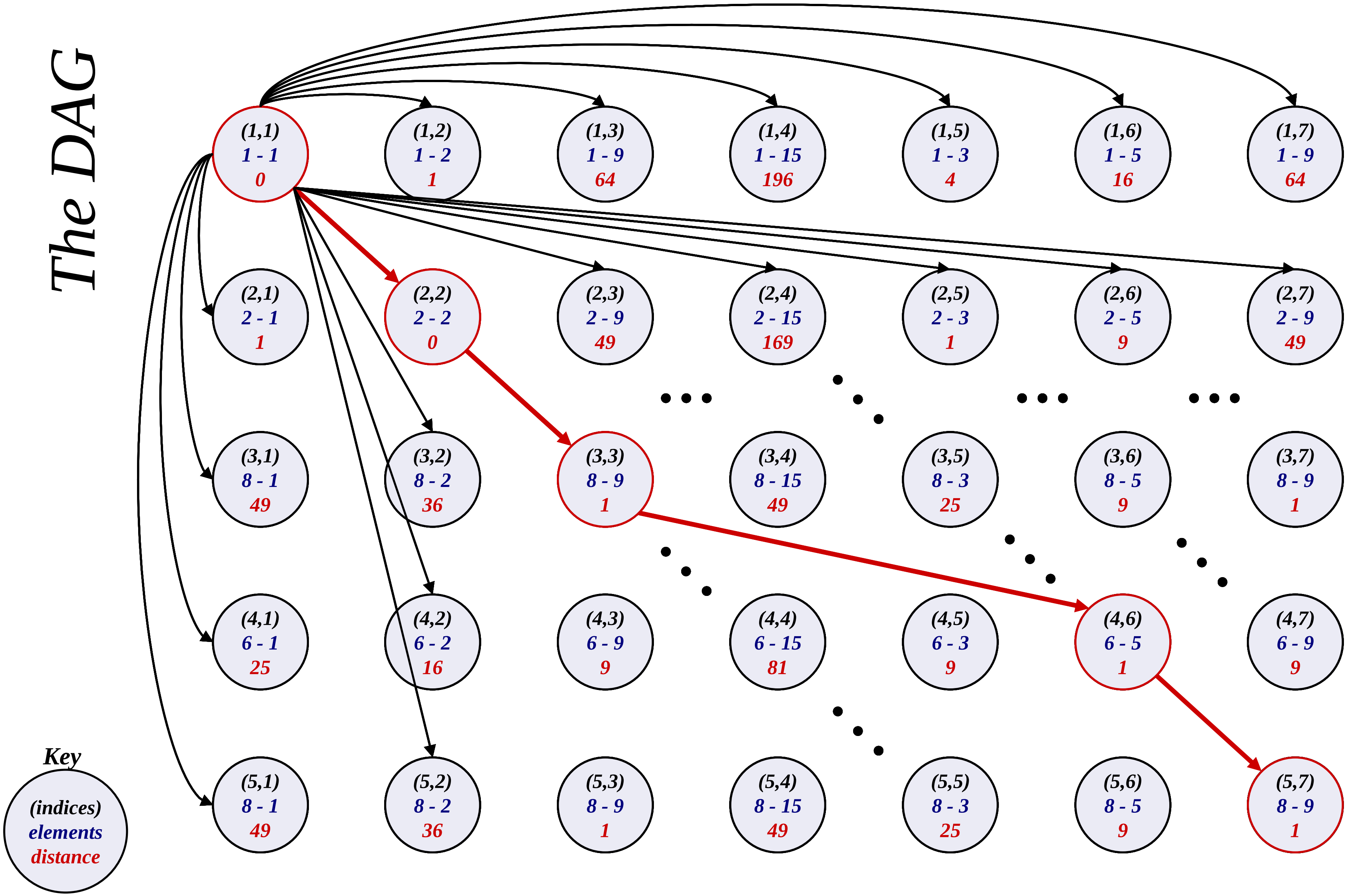
$$\mathbf{a} = \{1, 2, 8, 6, 8\}$$
$$\mathbf{b} = \{1, 2, 9, 15, 3, 5, 9\}$$

The dissimilarity matrix

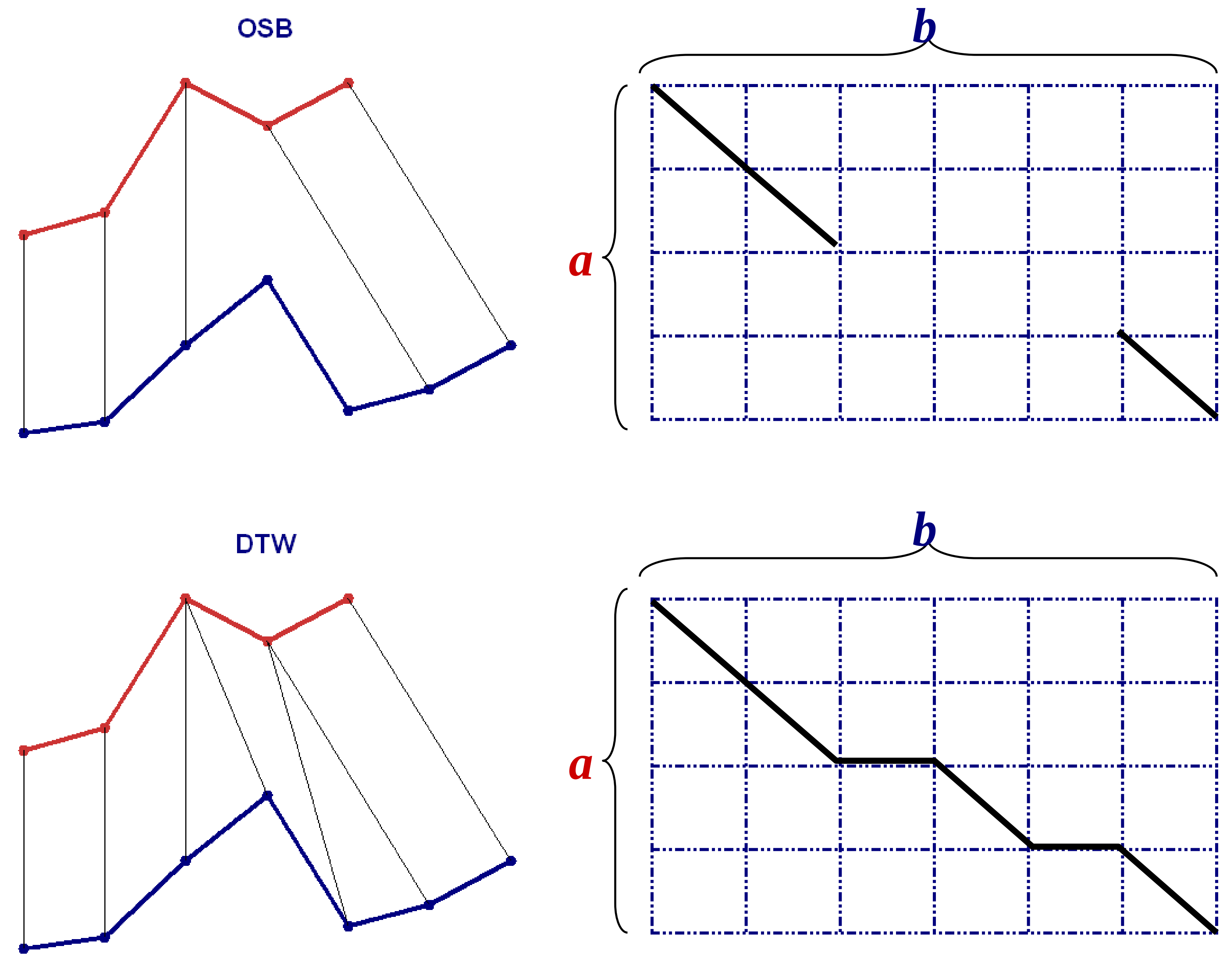
		b						
		1	**2**	**9**	**15**	**3**	**5**	**9**
a	**1**	0	1	64	196	4	16	64
	2	1	0	49	169	1	9	49
	8	49	36	1	49	25	9	1
	6	25	16	9	81	9	1	9
	8	49	36	1	49	25	9	1

$$d(a_i,b_j) = (a_i - b_j)^2$$

The DAG


(1,1) 1 - 1 0
(1,2) 1 - 2 1
(1,3) 1 - 9 64
(1,4) 1 - 15 196
(1,5) 1 - 3 4
(1,6) 1 - 5 16
(1,7) 1 - 9 64
(2,1) 2 - 1 1
(2,2) 2 - 2 0
(2,3) 2 - 9 49
(2,4) 2 - 15 169
(2,5) 2 - 3 1
(2,6) 2 - 5 9
(2,7) 2 - 9 49
(3,1) 8 - 1 49
(3,2) 8 - 2 36
(3,3) 8 - 9 1
(3,4) 8 - 15 49
(3,5) 8 - 3 25
(3,6) 8 - 5 9
(3,7) 8 - 9 1
(4,1) 6 - 1 25
(4,2) 6 - 2 16
(4,3) 6 - 9 9
(4,4) 6 - 15 81
(4,5) 6 - 3 9
(4,6) 6 - 5 1
(4,7) 6 - 9 9
(5,1) 8 - 1 49
(5,2) 8 - 2 36
(5,3) 8 - 9 1
(5,4) 8 - 15 49
(5,5) 8 - 3 25
(5,6) 8 - 5 9
(5,7) 8 - 9 1
Key
(indices)
elements
distance

The Final Result



Calculating the Jump Cost

- Given query *a* and a set of targets *B*
  1. $C(a,b) = \operatorname{mean}_i(\min_j(d(a_i,b_j))) + \operatorname{std}_i(\min_j(d(a_i,b_j)))$
  2. $C(a) = \operatorname{mean}\{C(a,b) : b \in B\}$
  3. Use a constant *C* found by training

		b						
		1	**2**	**9**	**15**	**3**	**5**	**9**
a	**1**	0	1	64	196	4	16	64
	2	1	0	49	169	1	9	49
	8	49	36	1	49	25	9	1
	6	25	16	9	81	9	1	9
	8	49	36	1	49	25	9	1

in dist for each a_i: 0, 0, 1, 1, 1

Mean = 0.6000

Std = 0.5477

Jumpcost = 1.1477

Thank you!

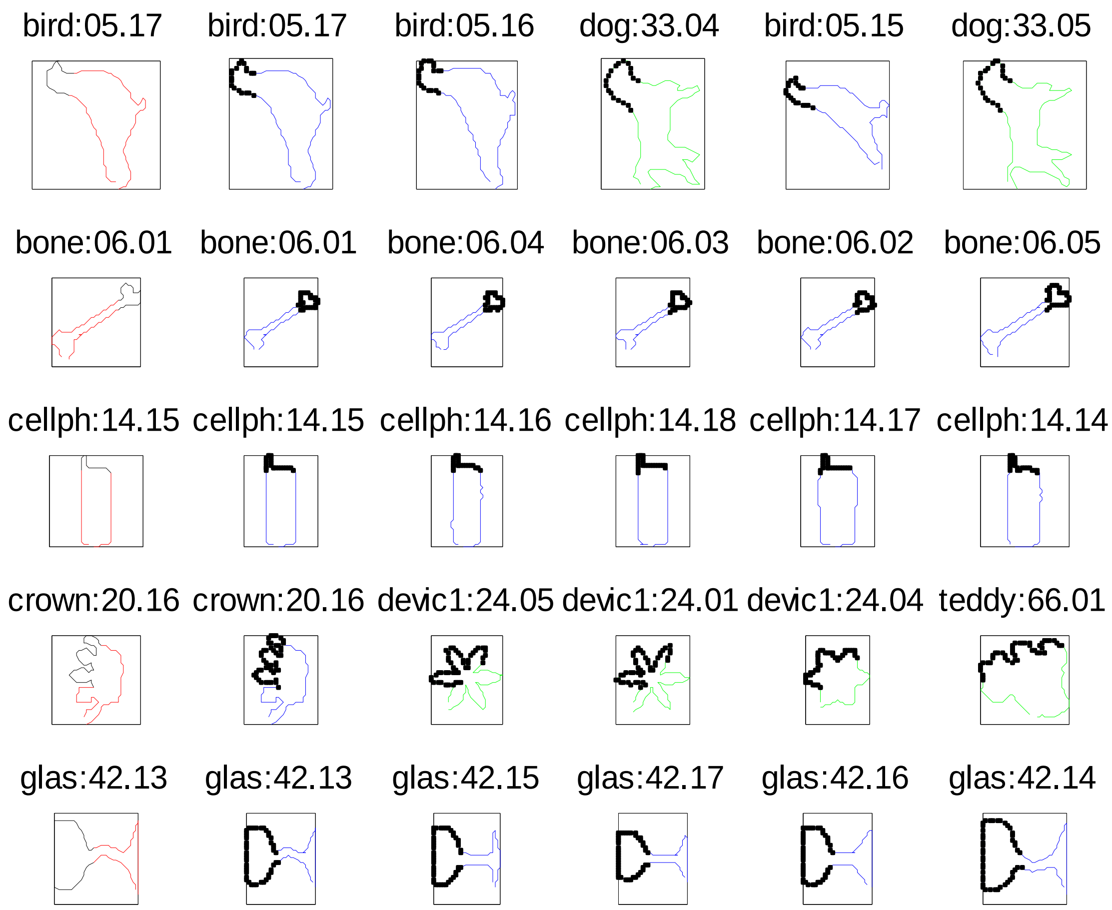
Any questions?

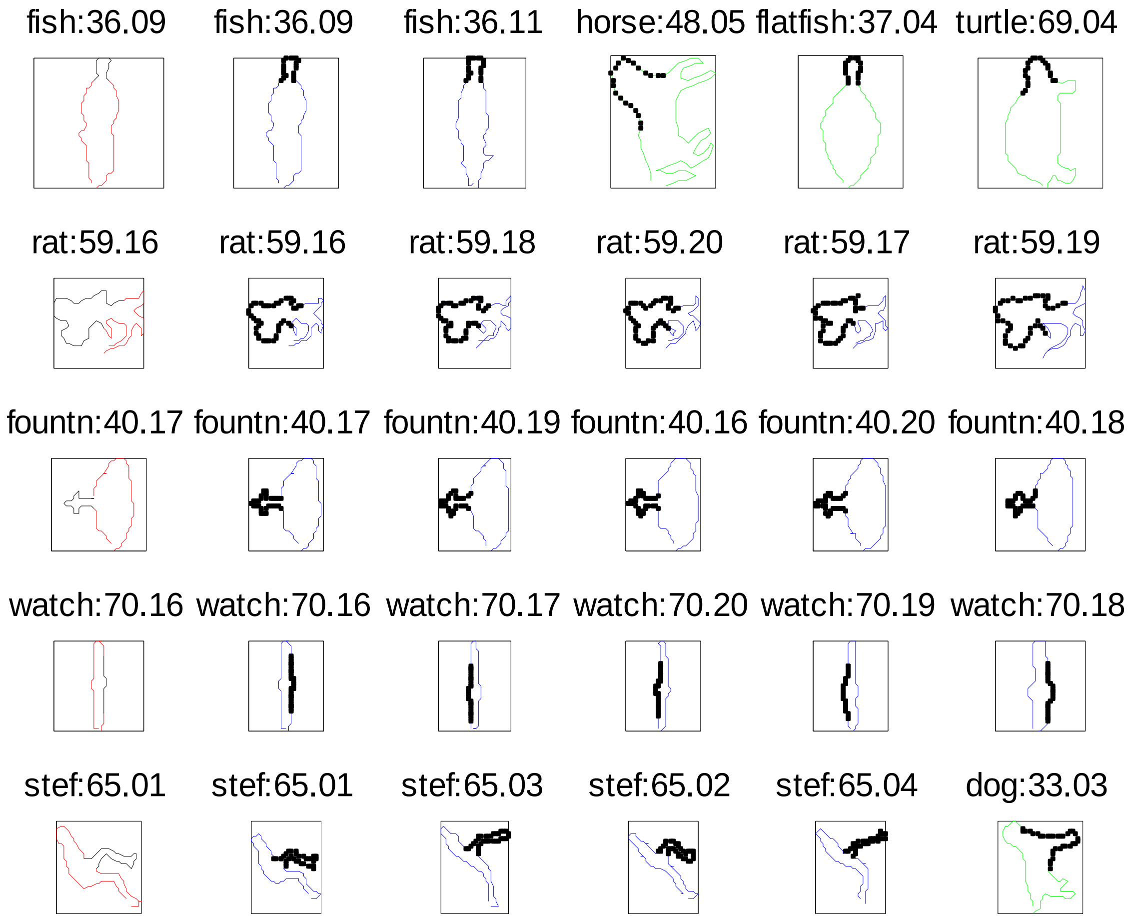
Appendix – Experimental Results

- UCR dataset results

Name	Number of Classes	Size of Training Set	Size of Testing Set	Time Series Length	Euclidean Distance Accuracy	DTW with Best Warping Window (r)	DTW without Warping Window	OSB
Synthetic Control	6	300	300	60	0.120	0.017 (6)	0.007	0.030
Gun-point	2	50	150	150	0.087	0.087 (0)	0.093	0.027
CBF	3	30	900	128	0.148	0.004 (11)	0.003	0.011
Face(all)	14	560	1690	131	0.286	0.192 (3)	0.192	0.111
OSU Leaf	6	200	242	427	0.483	0.384 (7)	0.409	0.409
Swedish Leaf	15	500	625	128	0.213	0.157 (2)	0.210	0.091
50Words	50	450	455	270	0.369	0.242 (6)	0.310	0.259
Trace	4	100	100	275	0.240	0.010 (3)	0.000	0.200
Two Patterns	4	1000	4000	128	0.090	0.002 (4)	0.000	0.000
Wafer	2	1000	6174	152	0.005	0.005 (1)	0.020	0.002
Face (four)	4	24	88	350	0.216	0.114 (2)	0.170	0.045
Lightening-2	2	60	61	637	0.246	0.131 (6)	0.131	0.148
Lightning-7	7	70	73	319	0.425	0.288 (5)	0.274	0.233
ECG	2	100	100	96	0.120	0.120 (0)	0.230	0.100
Adiac	37	390	391	176	0.389	0.391 (3)	0.396	0.386
Yoga	2	300	3000	426	0.170	0.155 (2)	0.164	0.150
Fish	7	175	175	463	0.217	0.160 (4)	0.167	0.103
Beef	5	30	30	470	0.467	0.467	0.500	0.467
Coffee	2	28	28	286	0.250	0.179	0.179	0.250
OliveOil	4	30	30	570	0.133	0.167	0.133	0.133

- MPEG 7 dataset





	OSB	DTW	DTWCW	LCSS
1NN	100%	0%	90%	90%
5NN	92%	2%	72%	42%
10NN	84%	2%	67%	34%
20NN	67%	3%	59%	26%