Representing Images; Detecting faces in images

Class 6. 17 Sep 2012

Instructor: Bhiksha Raj

Administrivia

- Project teams?
  - By the end of the month..
- Project proposals?
  - Please send proposals to Prasanna, and cc me.

Administrivia

- Basics of probability: Will not be covered
- Very nice lecture by Aarthi Singh
  - http://www.cs.cmu.edu/~epxing/Class/10701/Lecture/lecture2.pdf
- Another nice lecture by Paris Smaragdis
  - http://courses.engr.illinois.edu/cs598ps/CS598PS/Topics_and_Materials.html
    - Look for Lecture 2
- Amazing number of resources on the web
- Things to know:
  - Basic probability, Bayes rule
  - Probability distributions over discrete variables
  - Probability density and Cumulative density over continuous variables
    - Particularly Gaussian densities
  - Moments of a distribution
  - What is independence
  - Nice to know
    - What is maximum likelihood estimation
    - MAP estimation

Representing an Elephant

- It was six men of Indostan,
  To learning much inclined,
  Who went to see the elephant,
  (Though all of them were blind),
  That each by observation
  Might satisfy his mind.

- The first approached the elephant,
  And happening to fall
  Against his broad and sturdy side,
  At once began to bawl:
  "God bless me! But the elephant
  Is very like a wall!“

- The second, feeling of the tusk,
  Cried: "Ho! What have we here,
  So very round and smooth and sharp?
  To me 'tis very clear,
  This wonder of an elephant
  Is very like a spear!“

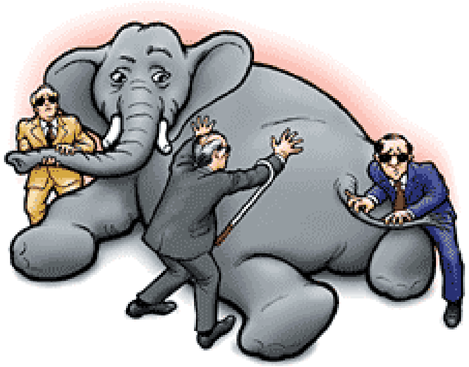
- The third approached the animal,
  And happening to take
  The squirming trunk within his hands,
  Thus boldly up and spake:
  "I see," quoth he, "the elephant
  Is very like a snake!“

- The fourth reached out an eager hand,
  And felt about the knee.
  "What most this wondrous beast is like
  Is might plain," quoth he;
  "Tis clear enough the elephant
  Is very like a tree."

- The fifth, who chanced to touch the ear,
  Said: "E'en the blindest man
  Can tell what this resembles most:
  Deny the fact who can,
  This marvel of an elephant
  Is very like a fan.“

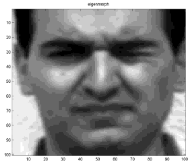
- The sixth no sooner had begun
  About the beast to grope,
  Than seizing on the swinging tail
  That fell within his scope,
  "I see," quoth he, "the elephant
  Is very like a rope.“

- And so these men of Indostan
  Disputed loud and long,
  Each in his own opinion
  Exceeding stiff and strong.
  Though each was partly right,
  All were in the wrong.



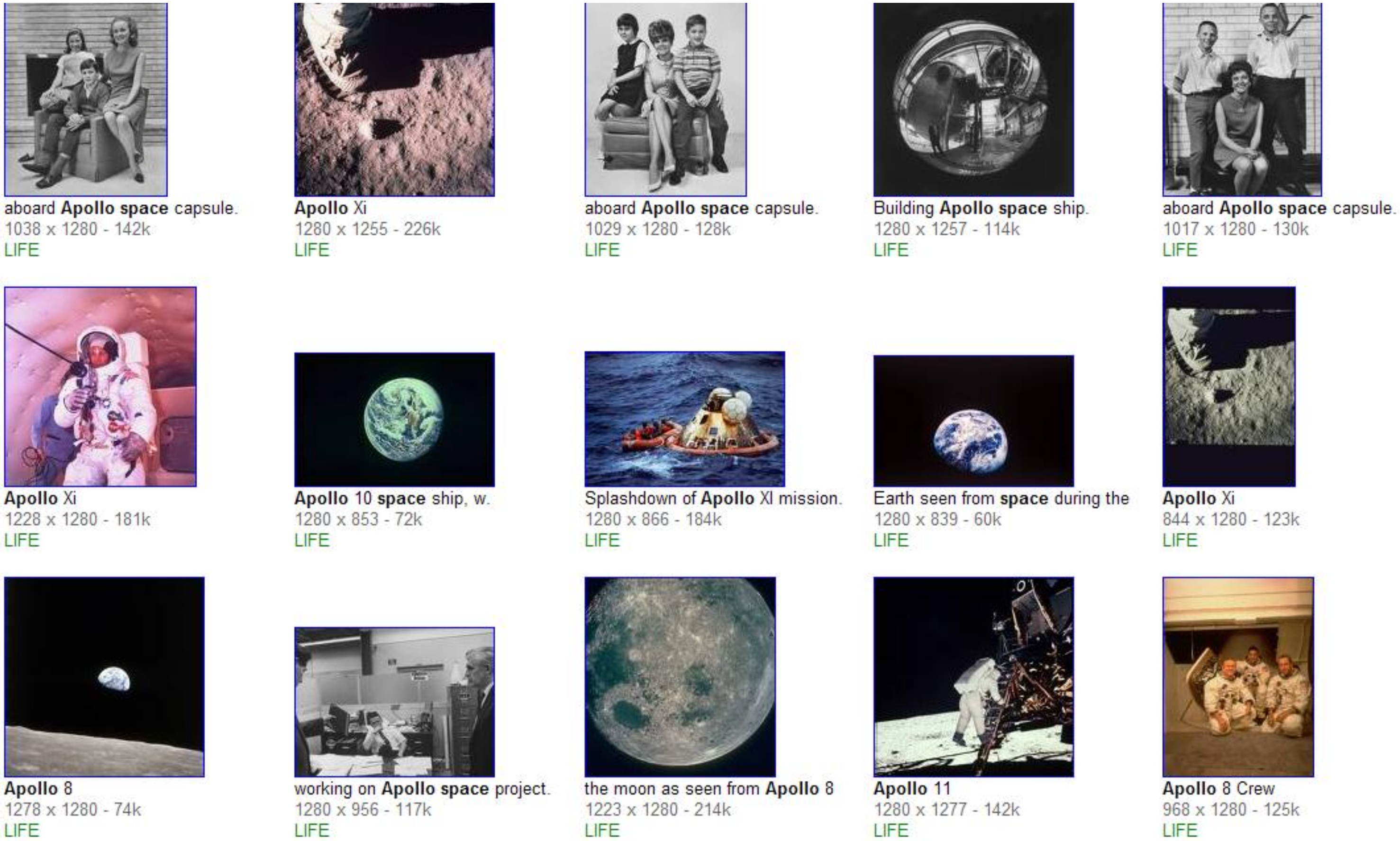
Representation

- Describe these images
  - Such that a listener can visualize what you are describing



- More images

Still more images



aboard **Apollo space** capsule.
1038 x 1280 - 142k
LIFE

Apollo Xi
1280 x 1255 - 226k
LIFE

aboard **Apollo space** capsule.
1029 x 1280 - 128k
LIFE

Building **Apollo space** ship.
1280 x 1257 - 114k
LIFE

aboard **Apollo space** capsule.
1017 x 1280 - 130k
LIFE

Apollo Xi
1228 x 1280 - 181k
LIFE

Apollo 10 **space** ship, w.
1280 x 853 - 72k
LIFE

Splashdown of **Apollo** XI mission.
1280 x 866 - 184k
LIFE

Earth seen from **space** during the
1280 x 839 - 60k
LIFE

Apollo Xi
844 x 1280 - 123k
LIFE

Apollo 8
1278 x 1280 - 74k
LIFE

working on **Apollo space** project.
1280 x 956 - 117k
LIFE

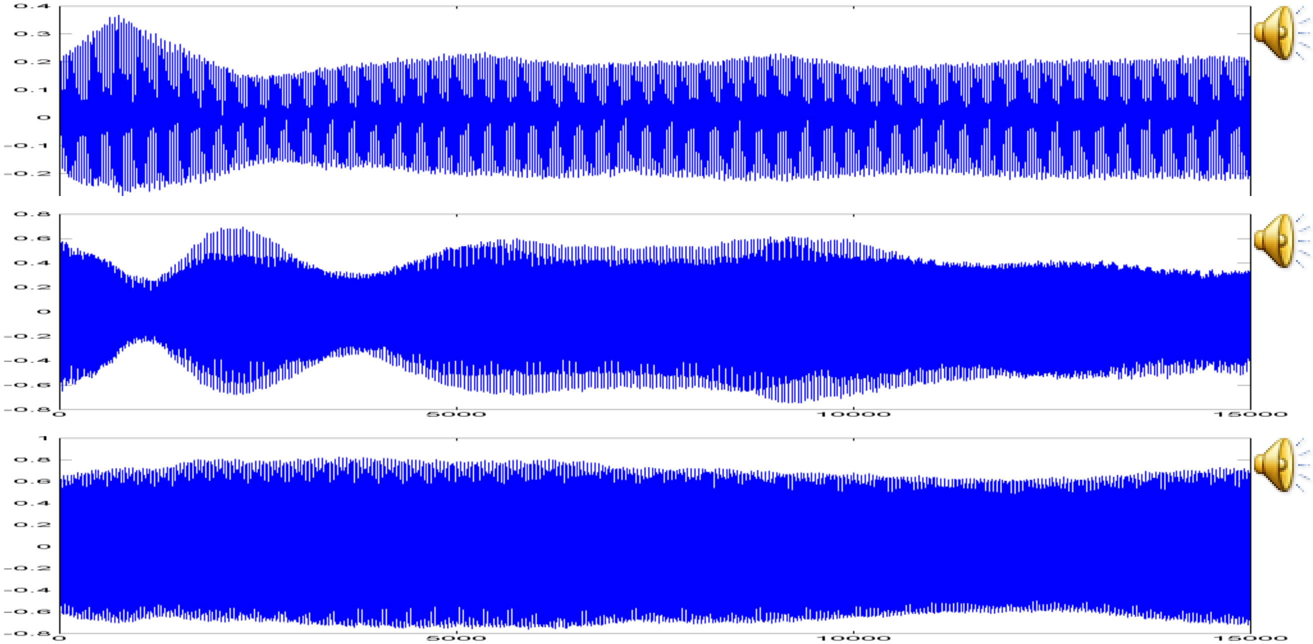
the moon as seen from **Apollo** 8
1223 x 1280 - 214k
LIFE

Apollo 11
1280 x 1277 - 142k
LIFE

Apollo 8 Crew
968 x 1280 - 125k
LIFE

How do you describe them?

Sounds

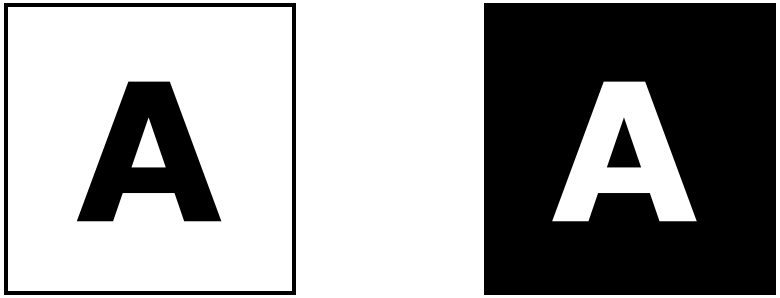


- Sounds are just sequences of numbers

- When plotted, they just look like blobs
  - Which leads to “natural sounds are blobs”
    - Or more precisely, “sounds are sequences of numbers that, when plotted, look like blobs”
  - Which wont get us anywhere

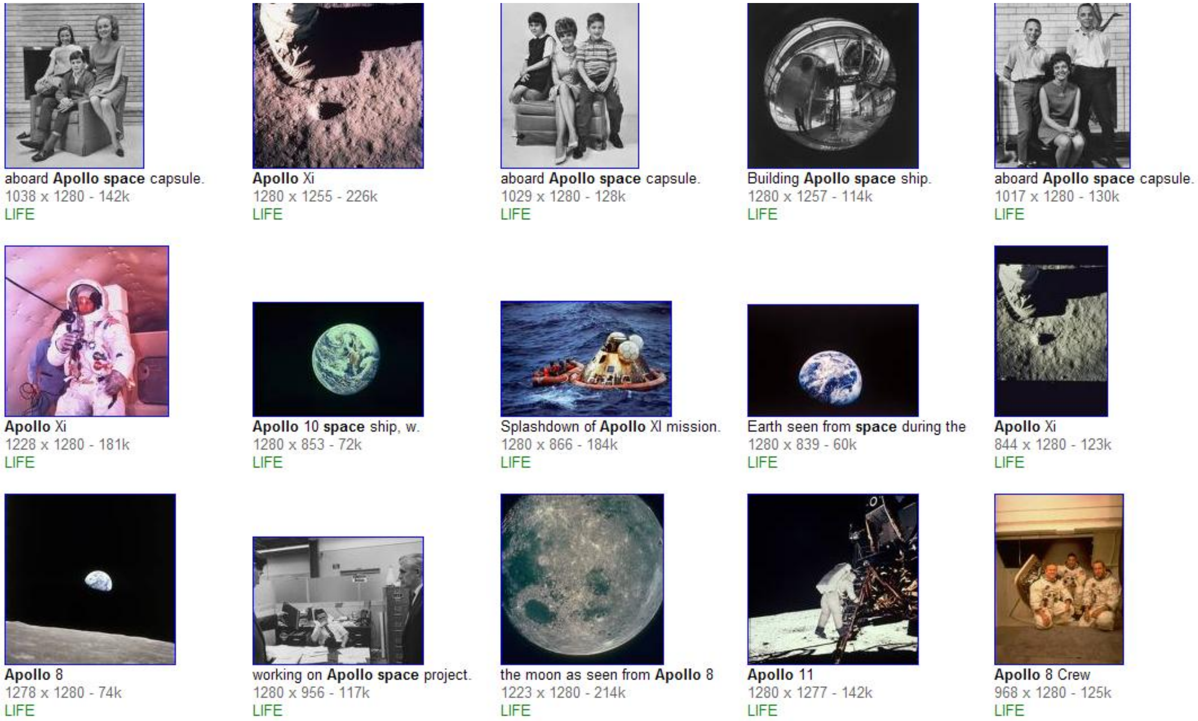
Representation

- Representation is description
- But in compact form
- Must describe the salient characteristics of the data
  - E.g. a pixel-wise description of the two images here will be completely different



- Must allow identification, comparison, storage, reconstruction..

Representing images



aboard **Apollo space** capsule.
1038 x 1280 - 142k
LIFE

Apollo Xi
1280 x 1255 - 226k
LIFE

aboard **Apollo space** capsule.
1029 x 1280 - 128k
LIFE

Building **Apollo space** ship.
1280 x 1257 - 114k
LIFE

aboard **Apollo space** capsule.
1017 x 1280 - 130k
LIFE

Apollo Xi
1228 x 1280 - 181k
LIFE

Apollo 10 **space** ship, w.
1280 x 853 - 72k
LIFE

Splashdown of **Apollo** XI mission.
1280 x 866 - 184k
LIFE

Earth seen from **space** during the
1280 x 839 - 60k
LIFE

Apollo Xi
844 x 1280 - 123k
LIFE

Apollo 8
1278 x 1280 - 74k
LIFE

working on **Apollo space** project.
1280 x 956 - 117k
LIFE

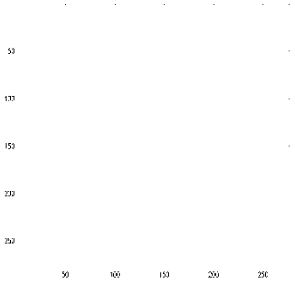
the moon as seen from **Apollo** 8
1223 x 1280 - 214k
LIFE

Apollo 11
1280 x 1277 - 142k
LIFE

Apollo 8 Crew
968 x 1280 - 125k
LIFE

- The most common element in the image: background
  - Or rather large regions of relatively featureless shading
  - Uniform sequences of numbers

Representing images using a “plain” image



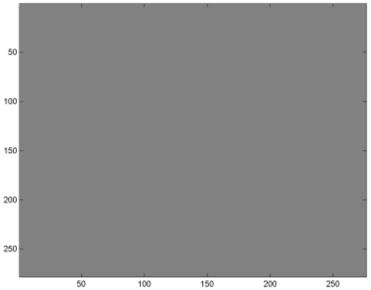
$$B = \begin{bmatrix} 1 \\ 1 \\ . \\ 1 \end{bmatrix} \qquad \text{Image} = \begin{bmatrix} pixel1 \\ pixel2 \\ . \\ pixel N \end{bmatrix}$$

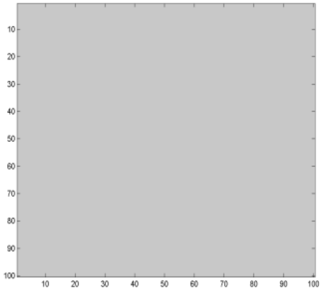
- Most of the figure is a more-or-less uniform shade
  - Dumb approximation – a image is a block of uniform shade
    - Will be mostly right!
  - How much of the figure is uniform?
- How? Projection
  - Represent the images as vectors and compute the projection of the image on the “basis”

$$BW \approx \text{Image}$$

$$W = pinv(B)\,\text{Image}$$

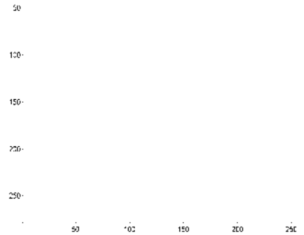
$$PROJECTION = BW = B(B^T B)^{-1} B^T.\text{Image}$$



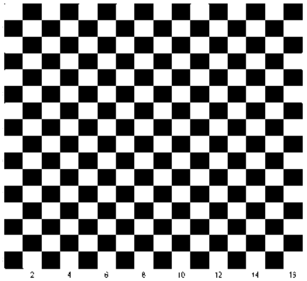


Adding more bases

B$_1$



B$_2$



$$\begin{array}{cc} & \begin{array}{cc} B_1 & B_2 \end{array} \\ B = & \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \end{array}$$

- Lets improve the approximation
- Images have some fast varying regions
  - Dramatic changes
  - Add a second picture that has very fast changes
    - A checkerboard where every other pixel is black and the rest are white

$$\text{Image} \approx w_1 B_1 + w_2 B_2$$

$$W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \qquad B = [B_1 \;\; B_2]$$

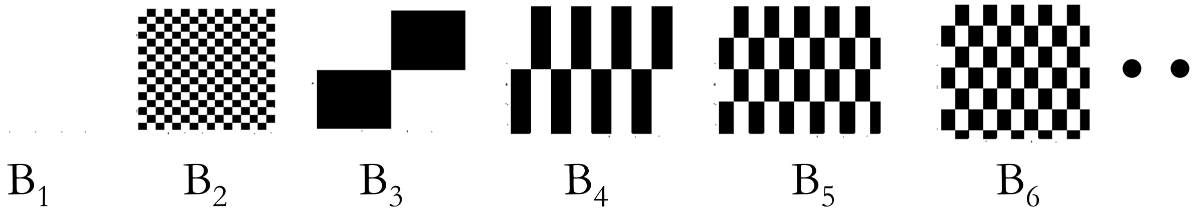
$$BW \approx \text{Image}$$

$$W = pinv(B)\text{Image}$$

$$PROJECTION = BW = B(B^T B)^{-1} B^T.\text{Image}$$

10
20
30
40
50
60
10
20
30
40
50
60

Adding still more bases



- Regions that change with different speeds

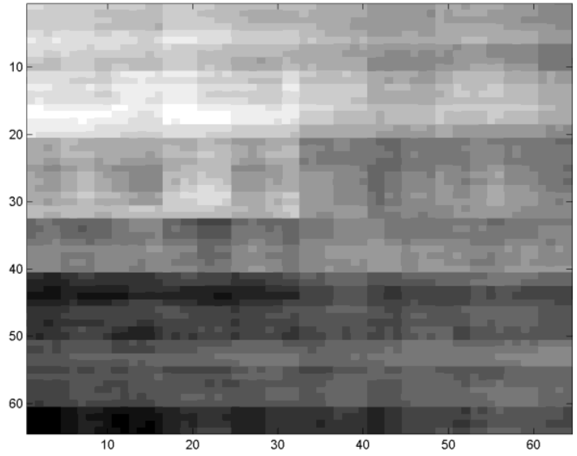
$$\mathrm{Image} \approx w_1 B_1 + w_2 B_2 + w_3 B_3 + \ldots$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \end{bmatrix} \qquad B = [B_1 \ \ B_2 \ \ B_3]$$

$$BW \approx \mathrm{Image}$$

$$W = pinv(B)\,\mathrm{Image}$$

$$PROJECTION = BW = B(B^T B)^{-1} B^T.\mathrm{Image}$$



Getting closer at 625 bases!

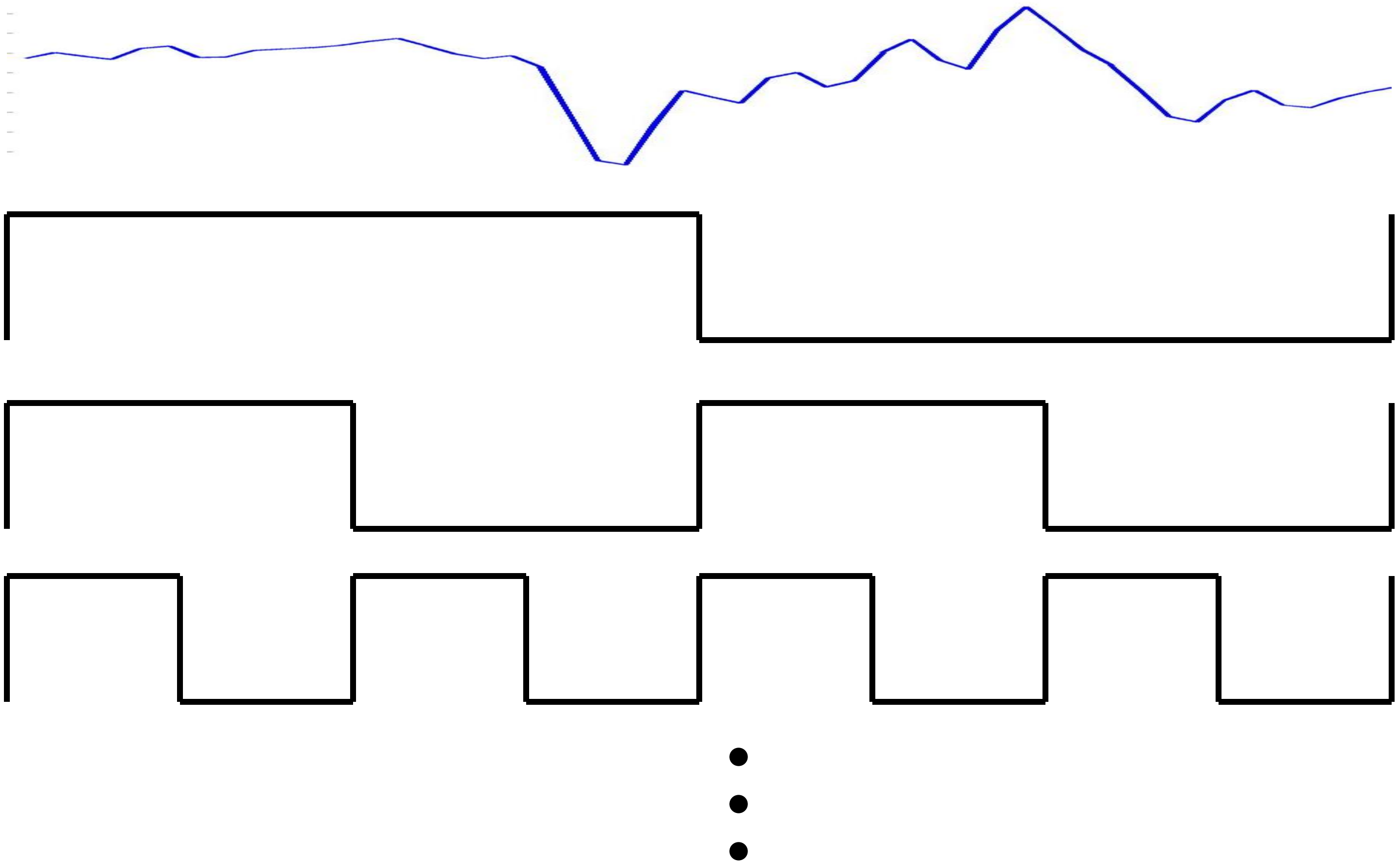
Representation using checkerboards

- A "standard" representation
  - Checker boards are the same regardless of what picture you're trying to describe
    - As opposed to using "nose shape" to describe faces and "leaf colour" to describe trees.

- Any image can be specified as (for example) 0.8*checkerboard(0) + 0.2*checkerboard(1) + 0.3*checkerboard(2) ..

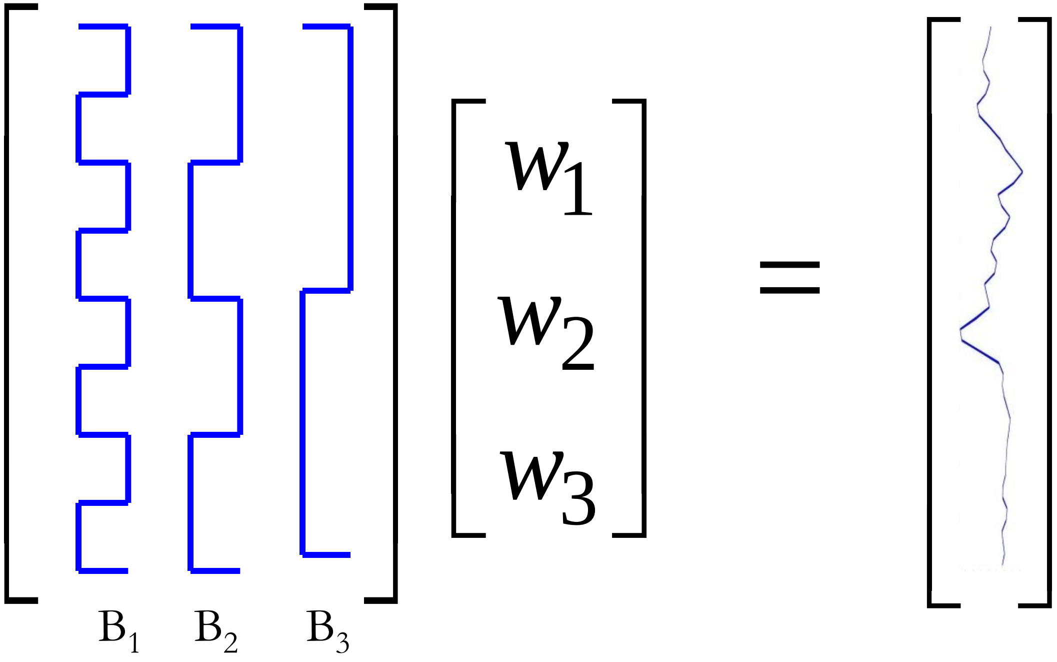
- The definition is sufficient to reconstruct the image to some degree
  - Not perfectly though

What about sounds?



- Square wave equivalents of checker boards

Projecting sounds



$$Signal \approx w_1 B_1 + w_2 B_2 + w_3 B_3$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \qquad B = [B_1 \;\; B_2 \;\; B_3]$$

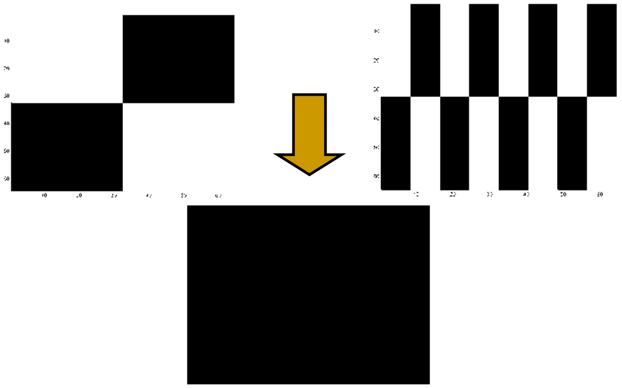
$$BW \approx Signal$$

$$W = pinv(B)Signal$$

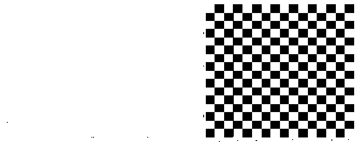
$$PROJECTION = BW = B(B^T B)^{-1} B.Signal$$

Why checkerboards are great bases

- We cannot explain one checkerboard in terms of another
  - The two are orthogonal to one another!



- This means that we can find out the contributions of individual bases separately
  - Joint decompostion with multiple bases with give us the same result as separate decomposition with each of them
  - This never holds true if one basis can explain another



$$B = \begin{array}{c} \begin{matrix} B_1 & B_2 \end{matrix} \\ \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix} \end{array}$$

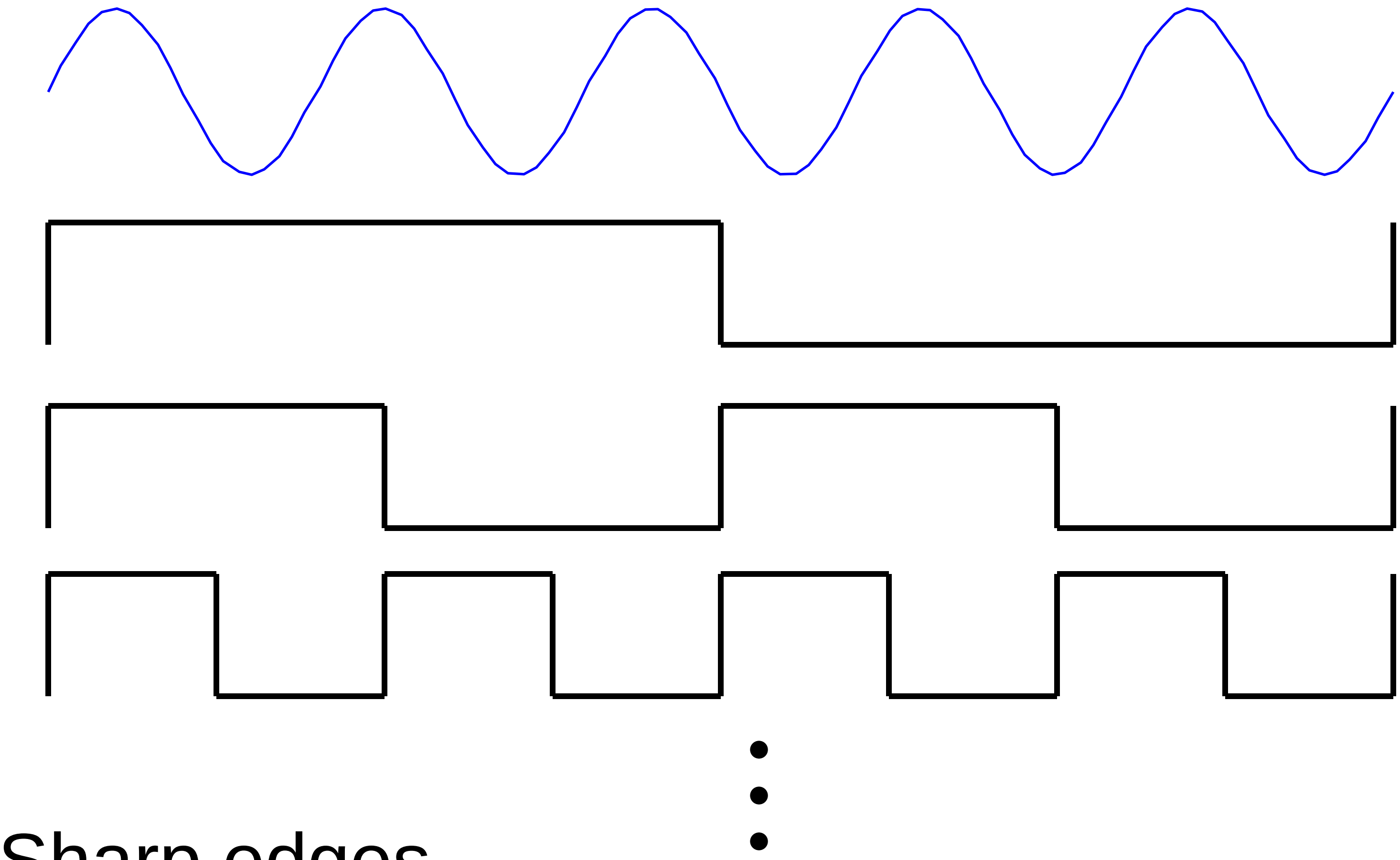
$$Image \approx w_1 B_1 + w_2 B_2$$

$$W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \qquad B = [B_1 \;\; B_2]$$

$$W = Pinv(B)\,Image$$

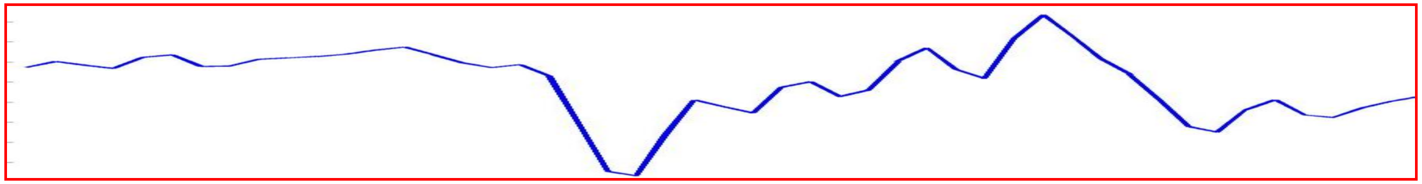
$$Pinv(B)\,Image = \begin{bmatrix} Pinv(B_1)\,Image \\ Pinv(B_2)\,Image \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

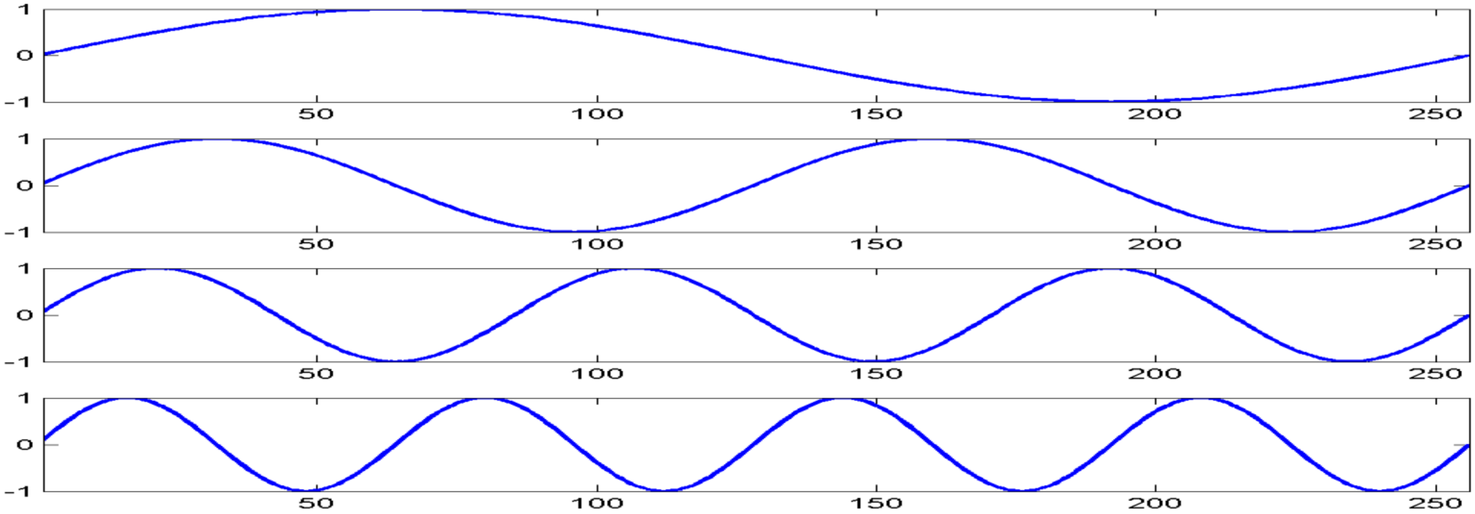
Checker boards are not good bases



- Sharp edges
  - Can *never* be used to explain rounded curves

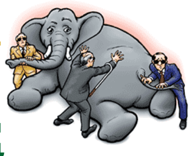
Sinusoids ARE good bases





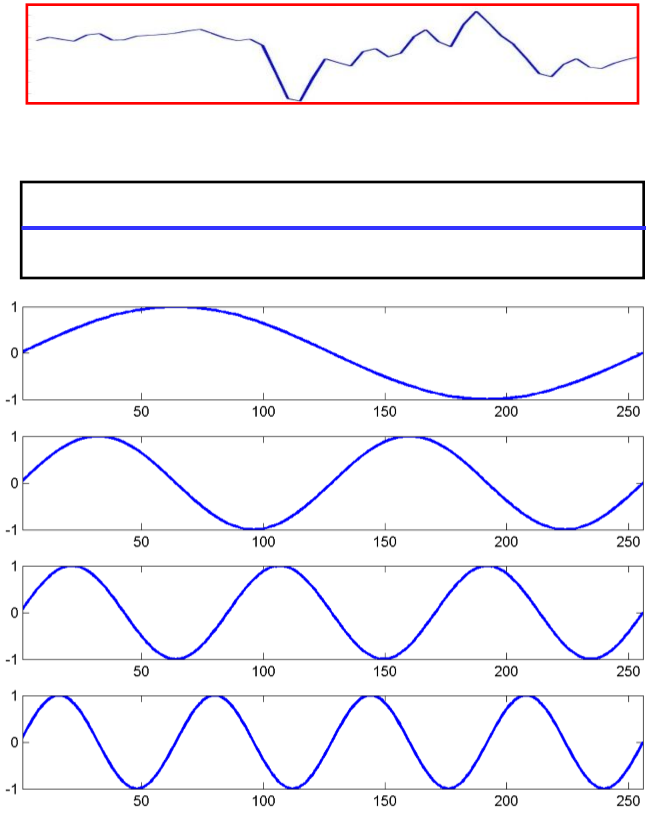
- They are orthogonal
- They can represent rounded shapes nicely
  - Unfortunately, they cannot represent sharp corners

What are the frequencies of the sinusoids

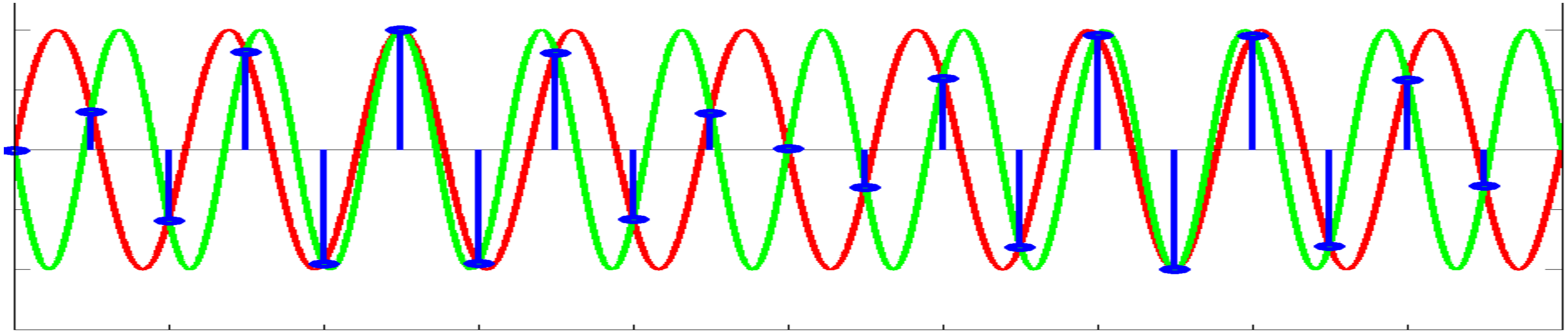


- Follow the same format as the checkerboard:
  - DC
  - The entire length of the signal is one period
  - The entire length of the signal is two periods.
  - And so on..

- The k-th sinusoid:
  - F(n) = sin(2πkn/L)
    - L is the length of the signal
    - k is the number of periods in L samples



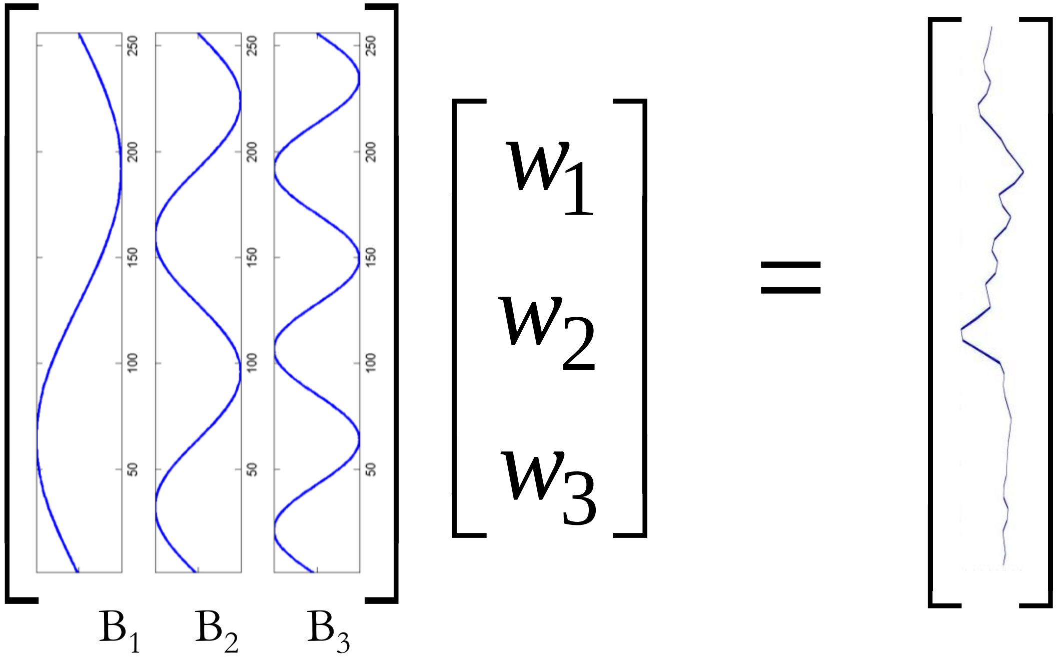
How many frequencies in all?



- A max of L/2 periods are possible
- If we try to go to (L/2 + X) periods, it ends up being identical to having (L/2 – X) periods
  - With sign inversion

- Example for L = 20
  - Red curve = sine with 9 cycles (in a 20 point sequence)
    - Y(n) = sin(2π9n/20)
  - Green curve = sine with 11 cycles in 20 points
    - Y(n) = -sin(2π11n/20)
  - The blue lines show the actual samples obtained
    - These are the only numbers stored on the computer
    - This set is the same for both sinusoids

How to compose the signal from sinusoids



$$Signal \approx w_1 B_1 + w_2 B_2 + w_3 B_3$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \qquad B = [B_1 \;\; B_2 \;\; B_3]$$

$$BW \approx Signal$$

$$W = pinv(B)Signal$$

$$PROJECTION = BW = B(B^T B)^{-1} B.Signal$$

- The sines form the vectors of the projection matrix
  - Pinv() will do the trick as usual

How to compose the signal from sinusoids

$$\begin{bmatrix} \sin(2\pi.0.0/\mathrm{L}) & \sin(2\pi.1.0/\mathrm{L}) & . & . & \sin(2\pi.(L/2).0/\mathrm{L}) \\ \sin(2\pi.0.1/\mathrm{L}) & \sin(2\pi.1.1/\mathrm{L}) & . & . & \sin(2\pi.(L/2).1/\mathrm{L}) \\ . & . & . & . & . \\ . & . & . & . & . \\ \sin(2\pi.0.(L-1)/\mathrm{L}) & \sin(2\pi.1.(L-1)/\mathrm{L}) & . & . & \sin(2\pi.(L/2).(L-1)/\mathrm{L}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_{L/2} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

L/2 columns only

$$Signal \approx w_1 B_1 + w_2 B_2 + w_3 B_3$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \qquad B = [B_1 \;\; B_2 \quad B_3]$$

$$Signal = \begin{bmatrix} s[0] \\ s[1] \\ . \\ s[L-1] \end{bmatrix}$$

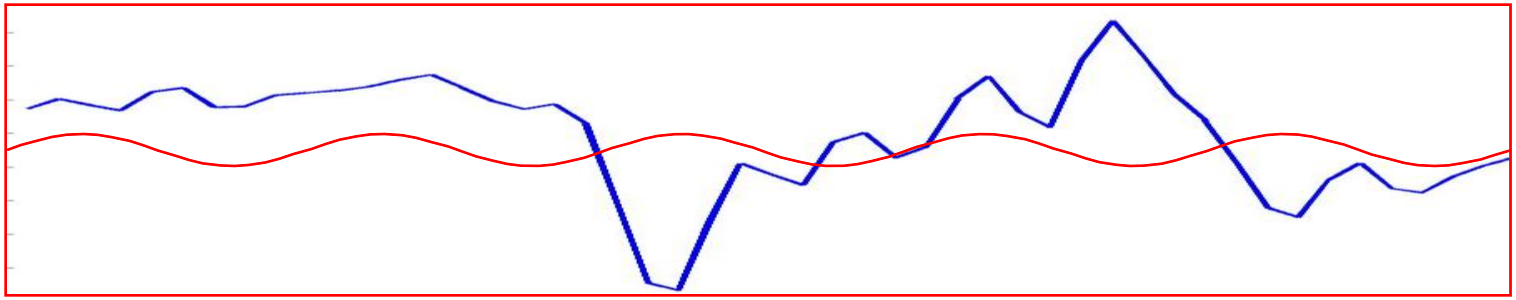
$$BW \approx Signal$$

$$W = pinv(B)Signal$$

$$PROJECTION = BW = B(B^T B)^{-1} B.Signal$$

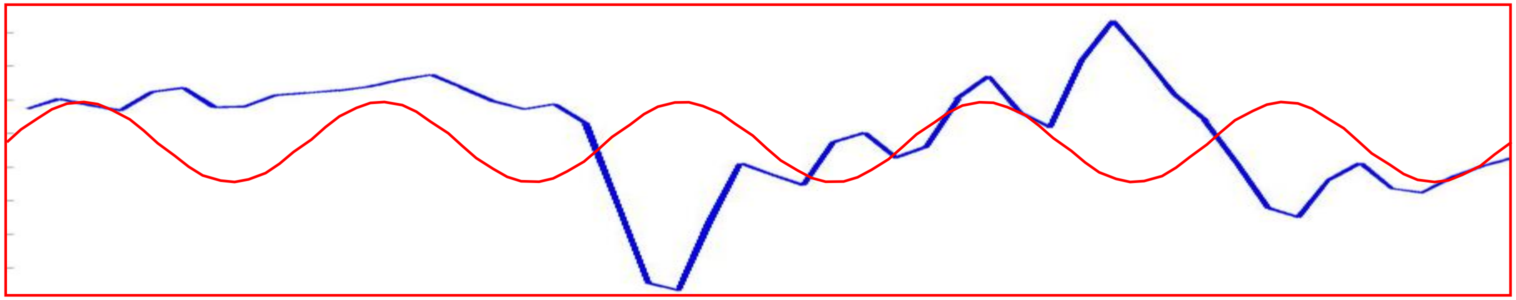
- The sines form the vectors of the projection matrix
  - Pinv() will do the trick as usual

Interpretation..



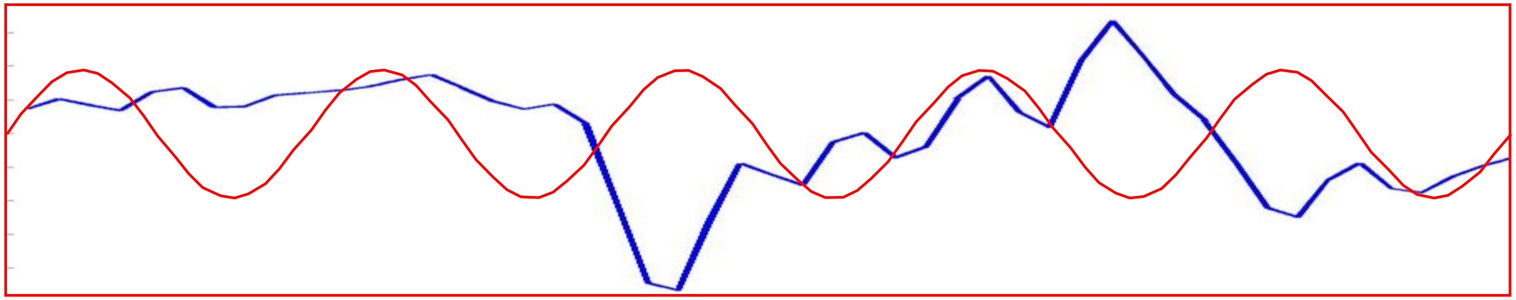
- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..



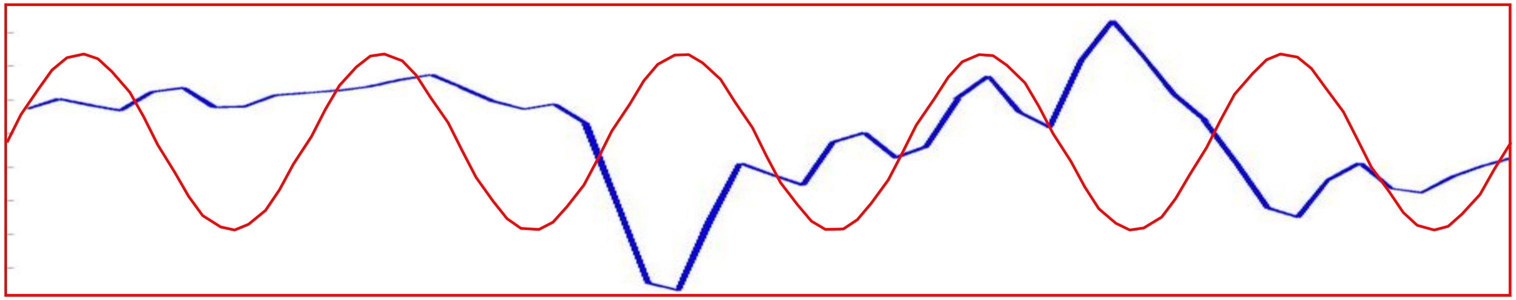
- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..



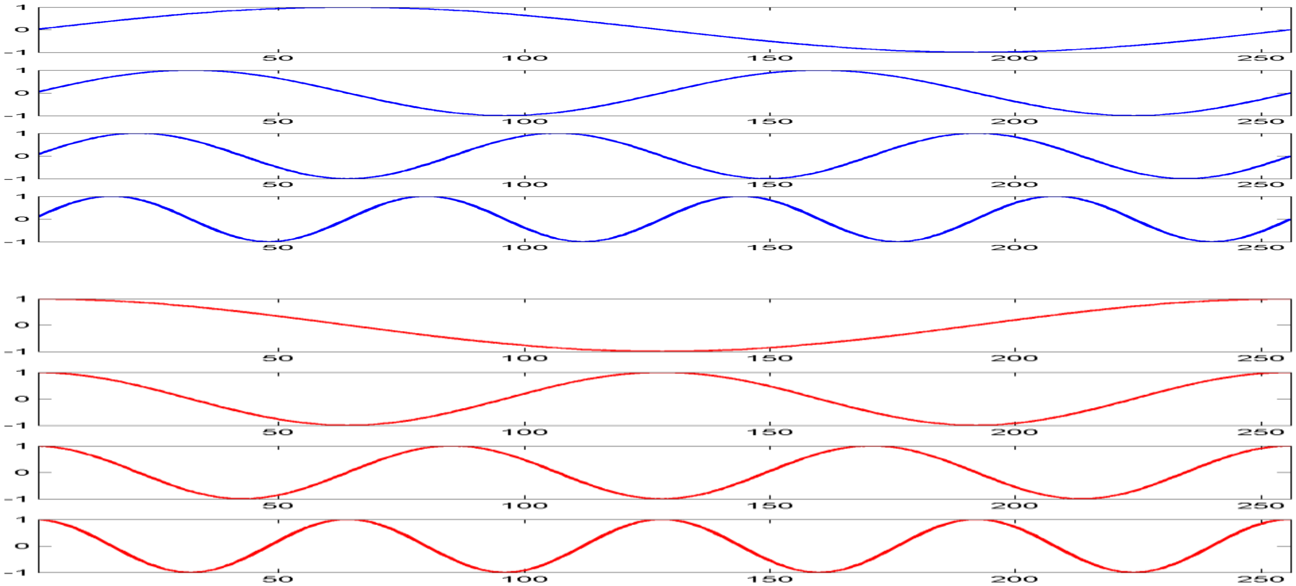
- Each sinusoid’s amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..



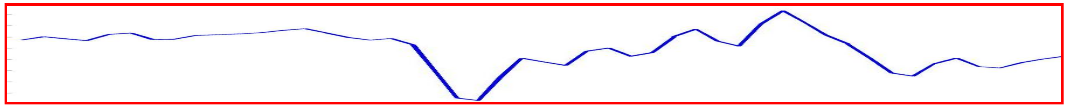
- Each sinusoid’s amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

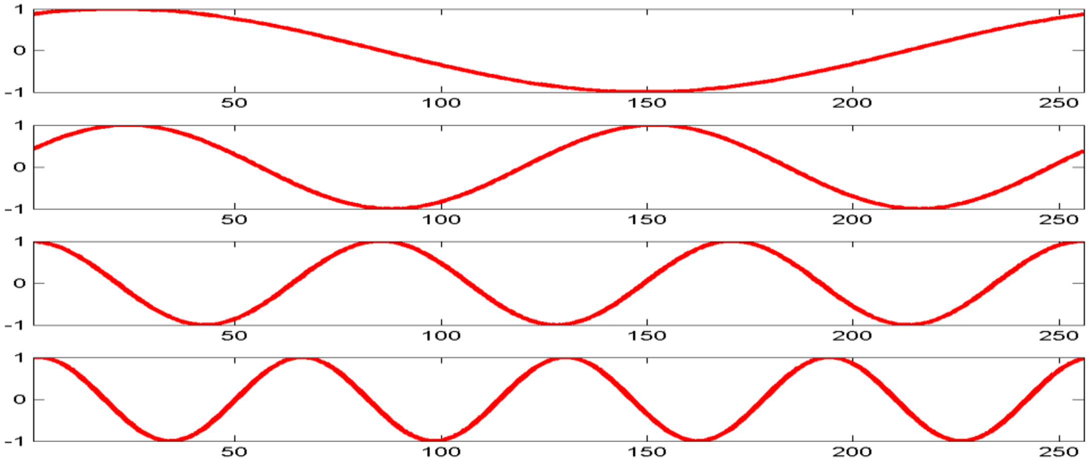
Sines by themselves are not enough



- Every sine starts at zero
  - Can never represent a signal that is non-zero in the first sample!
- Every cosine starts at 1
  - If the first sample is zero, the signal cannot be represented!

The need for phase





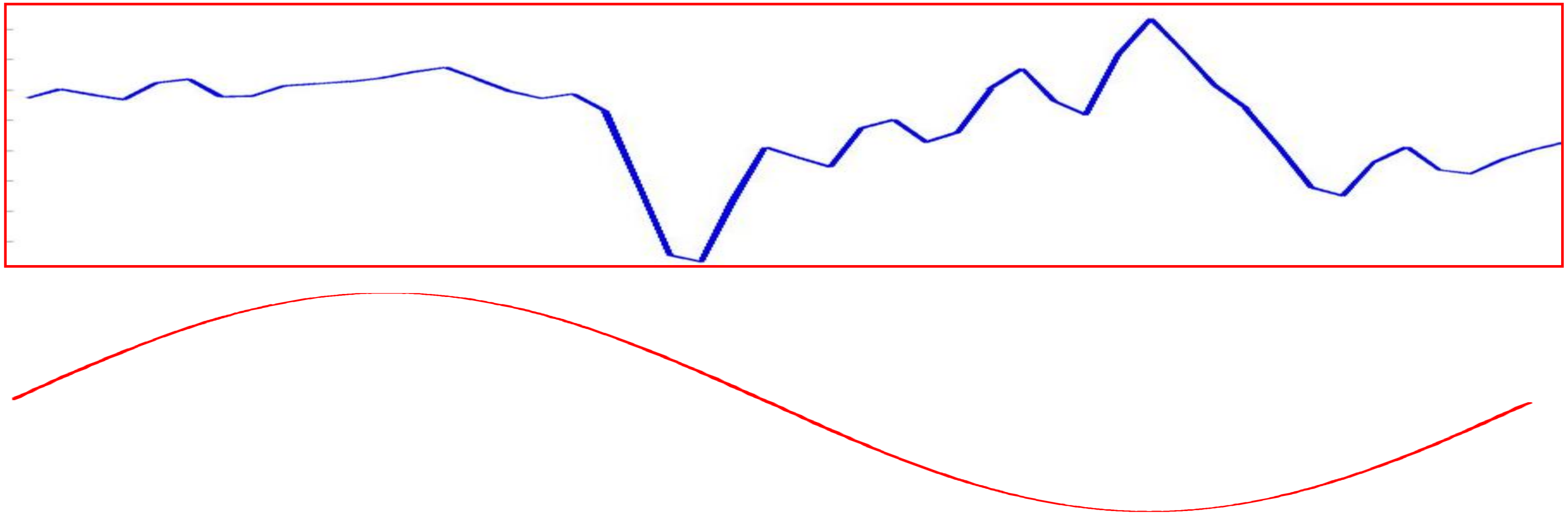
Sines are shifted: do not start with value = 0

- Allow the sinusoids to move!

$$signal = w_1 \sin(2\pi kn/N + \phi_1) + w_2 \sin(2\pi kn/N + \phi_2) + w_3 \sin(2\pi kn/N + \phi_3) + ...$$

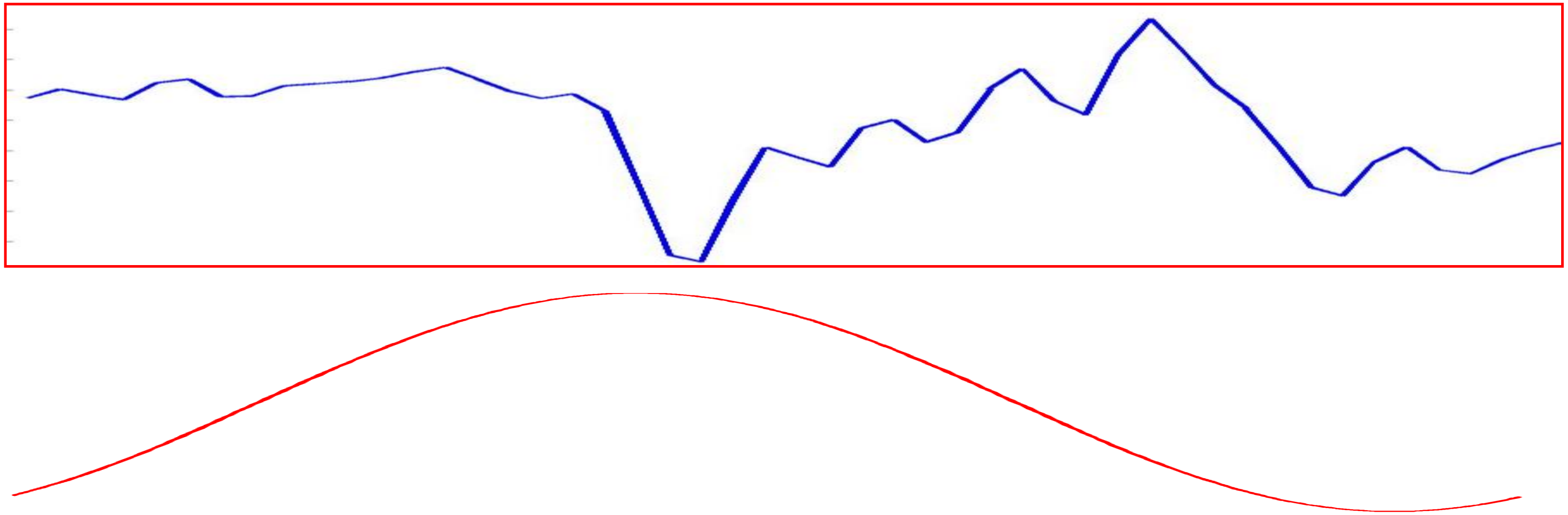
- How much do the sines shift?

Determining phase



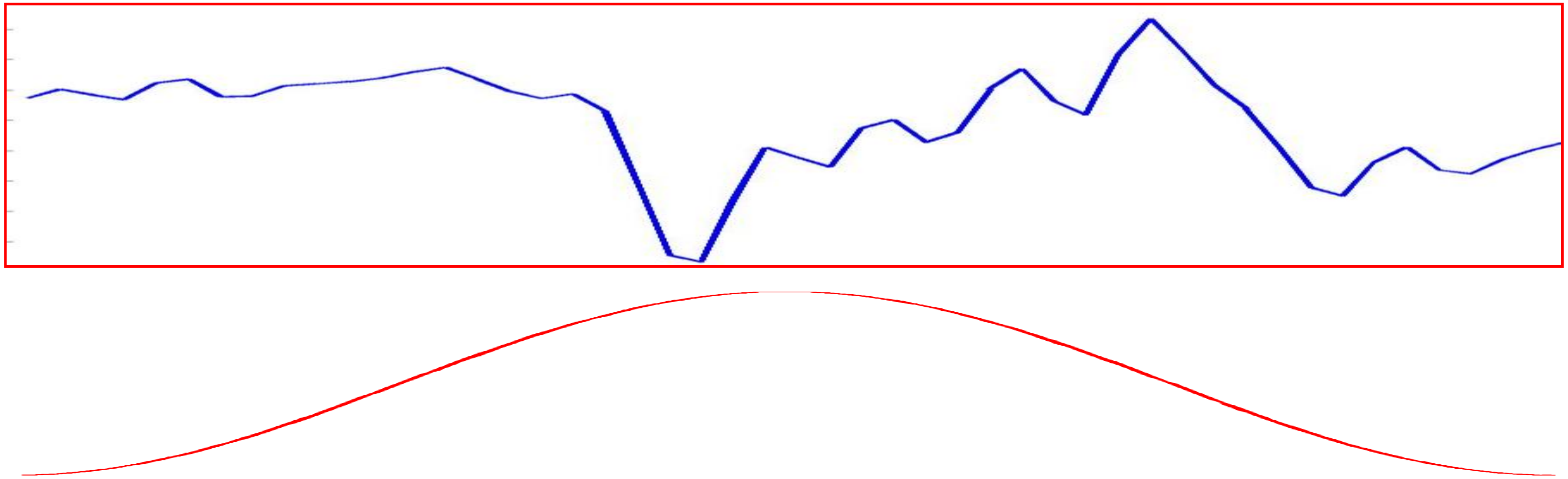
- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

Determining phase



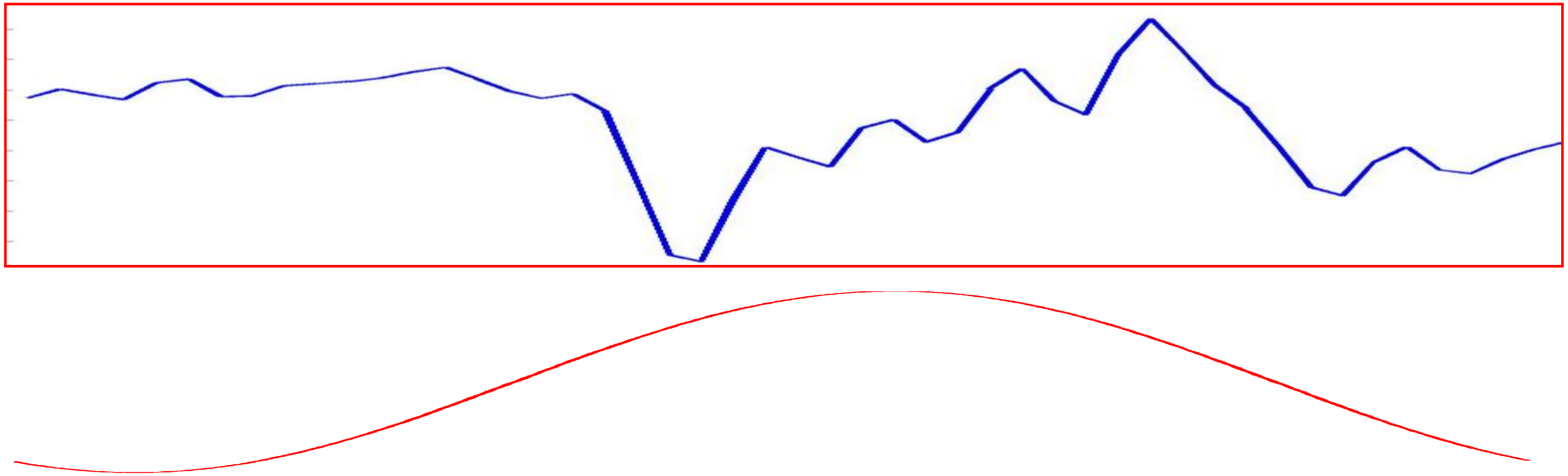
- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

Determining phase



- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

Determining phase



- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

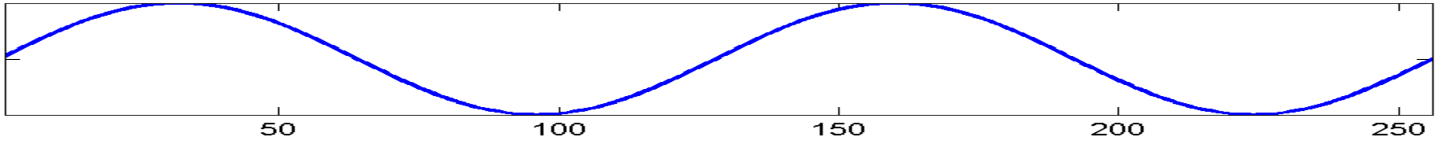
The problem with phase

$$\begin{bmatrix} \sin(2\pi.0.0/\mathrm{L}+\phi_0) & \sin(2\pi.1.0/\mathrm{L}+\phi_1) & . & . & \sin(2\pi.(L/2).0/\mathrm{L}+\phi_{\mathrm{L}/2}) \\ \sin(2\pi.0.1/\mathrm{L}+\phi_0) & \sin(2\pi.1.1/\mathrm{L}+\phi_1) & . & . & \sin(2\pi.(L/2).1/\mathrm{L}+\phi_{\mathrm{L}/2}) \\ . & . & . & . & . \\ . & . & . & . & . \\ \sin(2\pi.0.(L-1)/\mathrm{L}+\phi_0) & \sin(2\pi.1.(L-1)/\mathrm{L}+\phi_1) & . & . & \sin(2\pi.(L/2).(L-1)/\mathrm{L}+\phi_{\mathrm{L}/2}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_{L/2} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

- This can no longer be expressed as a simple linear algebraic equation
  - The phase is integral to the bases
    - I.e. there’s a component of the basis itself that must be estimated!
- Linear algebraic notation can only be used if the bases are *fully* known
  - *We can only (pseudo) invert a known matrix*

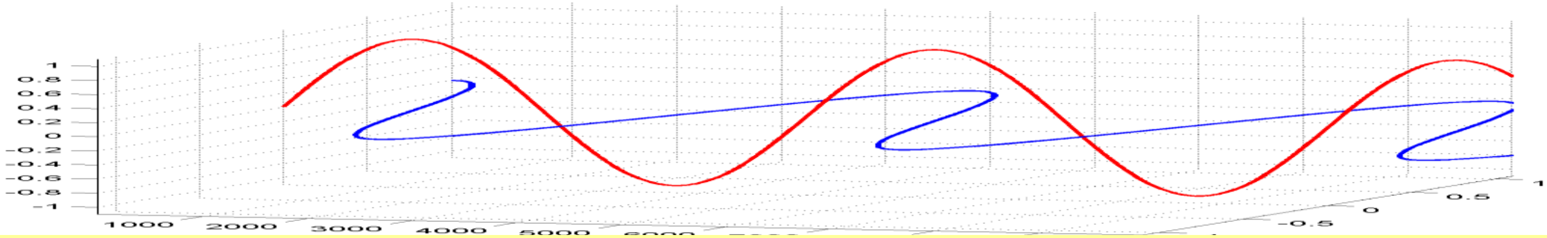
Complex Exponential to the rescue

$$b[n] = \sin(freq * n)$$



$$b_{freq}[n] = \exp(j * freq * n) = \cos(freq * n) + j\sin(freq * n)$$

$$j = \sqrt{-1}$$



$$\exp(j * freq * n + \phi) = \exp(j * freq * n)\exp(\phi) = \cos(freq * n + \phi) + j\sin(freq * n + \phi)$$

- The cosine is the real part of a complex exponential
  - The sine is the imaginary part
- A phase term for the sinusoid becomes a multiplicative term for the complex exponential!!

Complex Exponents to handle phase

$$\begin{bmatrix} \exp(j2\pi.0.0/L + j\phi_0) & \exp(j2\pi.1.0/L + j\phi_1) & . & . & \exp(j2\pi.(L-1).0/L + j\phi_{L-1}) \\ \exp(j2\pi.0.1/L + j\phi_0) & \exp(j2\pi.1.1/L + j\phi_1) & . & . & \exp(j2\pi.(L-1).1/L + j\phi_{L-1}) \\ . & . & . & . & . \\ . & . & . & . & . \\ \exp(j2\pi.0.(L-1)/L + j\phi_0) & \exp(j2\pi.1.(L-1)/L + j\phi_1) & . & . & \exp(j2\pi.(L-1).(L-1)/L + j\phi_{L-1}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

$$\begin{bmatrix} \exp(j2\pi.0.0/L)\exp(j\phi_0) & \exp(j2\pi.1.0/L)\exp(j\phi_1) & . & . & \exp(j2\pi.(L-1).0/L)\exp(j\phi_{L-1}) \\ \exp(j2\pi.0.1/L)\exp(j\phi_0) & \exp(j2\pi.1.1/L)\exp(j\phi_1) & . & . & \exp(j2\pi.(L-1).1/L)\exp(j\phi_{L-1}) \\ . & . & . & . & . \\ . & . & . & . & . \\ \exp(j2\pi.0.(L-1)/L)\exp(j\phi_0) & \exp(j2\pi.1.(L-1)/L)\exp(j\phi_1) & . & . & \exp(j2\pi.(L-1).(L-1)/L)\exp(j\phi_{L-1}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

$$\begin{bmatrix} \exp(j2\pi.0.0/L) & \exp(j2\pi.1.0/L) & . & . & \exp(j2\pi.(L-1).0/L) \\ \exp(j2\pi.0.1/L) & \exp(j2\pi.1.1/L)\exp(j\phi_1) & . & . & \exp(j2\pi.(L-1).1/L) \\ . & . & . & . & . \\ . & . & . & . & . \\ \exp(j2\pi.0.(L-1)/L) & \exp(j2\pi.1.(L-1)/L) & . & . & \exp(j2\pi.(L-1).(L-1)/L) \end{bmatrix} \begin{bmatrix} w_1\exp(j\phi_0) \\ w_2\exp(j\phi_1) \\ . \\ . \\ w_{L-1}\exp(j\phi_{L-1}) \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

Converts a non-linear operation into a linear algebraic operation!!

Complex exponentials are well behaved

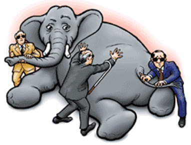
- Like sinusoids, a complex exponential of one frequency can never explain one of another
  - They are orthogonal
- They represent smooth transitions
- Bonus: They are *complex*
  - Can even model complex data!
- They can also model real data
  - exp(j x) + exp(-j x) is real
    - cos(x) + j sin(x) + cos(x) – j sin(x) = 2cos(x)

Complex Exponential Bases: Algebraic Formulation

$$\begin{bmatrix} \exp(j2\pi.0.0/L) & . & \exp(j2\pi.(L/2).0/L) & . & \exp(j2\pi.(L-1).0/L) \\ \exp(j2\pi.0.1/L) & . & \exp(j2\pi.(L/2).1/L) & . & \exp(j2\pi.(L-1).1/L) \\ . & . & . & . & . \\ . & . & . & . & . \\ \exp(j2\pi.0.(L-1)/L) & . & \exp(j2\pi.(L/2).(L-1)/L) & . & \exp(j2\pi.(L-1).(L-1)/L) \end{bmatrix} \begin{bmatrix} S_0 \\ . \\ S_{L/2} \\ . \\ S_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

- Note that $S_{L/2+x}$ = conjugate($S_{L/2-x}$) for real s

Shorthand Notation



$$W_L^{k,n} = \frac{1}{\sqrt{L}}\exp(j2\pi kn/L) = \frac{1}{\sqrt{L}}\left(\cos(2\pi kn/L) + j\sin(2\pi kn/L)\right)$$

$$\begin{bmatrix} W_L^{0,0} & . & W_L^{L/2,0} . & . & W_L^{L-1,0} \\ W_L^{0,1} & . & W_L^{L/2,1} . & . & W_L^{L-1,1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{0,L-1} & . & W_L^{L/2,L-1} & . & W_L^{L-1,L-1} \end{bmatrix} \begin{bmatrix} S_0 \\ . \\ S_{L/2} \\ . \\ S_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

- Note that $S_{L/2+x}$ = conjugate($S_{L/2-x}$)

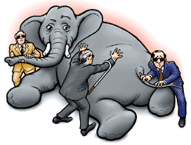
A quick detour

- Real Orthonormal matrix:
  - $XX^T = X X^T = I$
    - But only if all entries are real
  - The inverse of X is its own transpose

- Definition: Hermitian
  - X^H = Complex conjugate of X^T
    - Conjugate of a number $a + ib = a - ib$
    - Conjugate of $\exp(ix) = \exp(-ix)$

- Complex Orthonormal matrix
  - $XX^H = X^H X = I$
  - The inverse of a complex orthonormal matrix is its own Hermitian

$W^{-1} = W^{H}$



$$W = \begin{bmatrix} W_L^{0,0} & . & W_L^{L/2,0}. & . & W_L^{L-1,0} \\ W_L^{0,1} & . & W_L^{L/2,1}. & . & W_L^{L-1,1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{0,L-1} & . & W_L^{L/2,L-1} & . & W_L^{L-1,L-1} \end{bmatrix}$$

$$W_L^{k,n} = \frac{1}{\sqrt{L}} \exp(j2\pi kn/L)$$

$$W_L^{-k,n} = \frac{1}{\sqrt{L}} \exp(-j2\pi kn/L)$$

$$W^H = \begin{bmatrix} W_L^{0,0} & . & W_L^{-0,L/2}. & . & W_L^{-0,L-1} \\ W_L^{-1,0,} & . & W_L^{-1,L/2}. & . & W_L^{-1,L-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{-(L-1),0} & . & W_L^{-(L-1),L/2} & . & W_L^{-(L-1),(L-1)} \end{bmatrix}$$

- The complex exponential basis is orthonormal
  - Its inverse is its own Hermitian
  - $W^{-1} = W^{H}$

Doing it in matrix form

$$\begin{bmatrix} W_L^{0,0} & . & W_L^{L/2,0}. & . & W_L^{L-1,0} \\ W_L^{0,1} & . & W_L^{L/2,1}. & . & W_L^{L-1,1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{0,L-1} & . & W_L^{L/2,L-1} & . & W_L^{L-1,L-1} \end{bmatrix} \begin{bmatrix} S_0 \\ . \\ S_{L/2} \\ . \\ S_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

$$\begin{bmatrix} S_0 \\ . \\ S_{L/2} \\ . \\ S_{L-1} \end{bmatrix} = \begin{bmatrix} W_L^{0,0} & . & W_L^{-0,L/2}. & . & W_L^{-0,L-1} \\ W_L^{-1,0,} & . & W_L^{-1,L/2}. & . & W_L^{-1,L-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{-(L-1),0} & . & W_L^{-(L-1),L/2} & . & W_L^{-(L-1),(L-1)} \end{bmatrix} \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

- Because $W^{-1} = W^H$

The Discrete Fourier Transform

$$\begin{bmatrix} S_0 \\ . \\ S_{L/2} \\ . \\ S_{L-1} \end{bmatrix} = \begin{bmatrix} W_L^{0,0} & . & W_L^{-0,L/2}. & . & W_L^{-0,L-1} \\ W_L^{-1,0,} & . & W_L^{-1,L/2}. & . & W_L^{-1,L-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{-(L-1),0} & . & W_L^{-(L-1),L/2} & . & W_L^{-(L-1),(L-1)} \end{bmatrix} \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

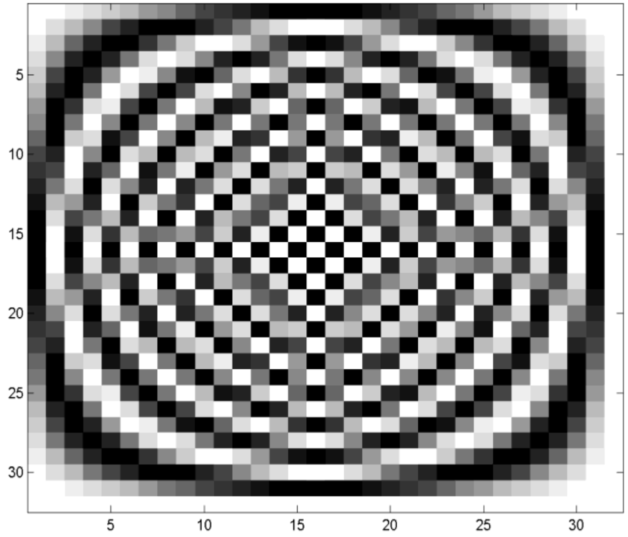
- The matrix to the right is called the “Fourier Matrix”
- The weights (S_0, S_1. . Etc.) are called the Fourier transform

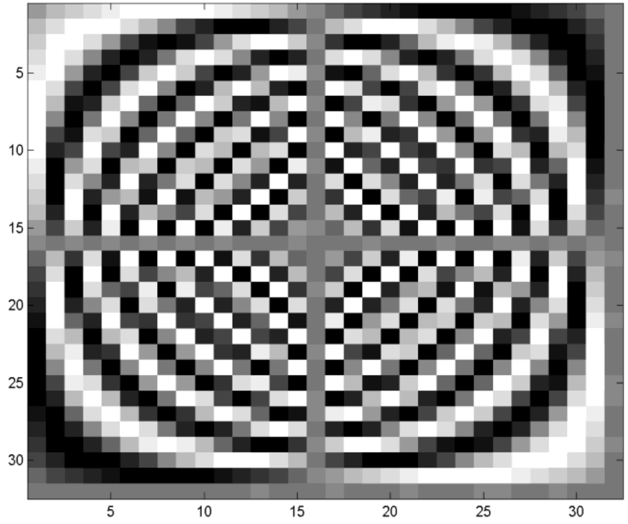
The Inverse Discrete Fourier Transform

$$\begin{bmatrix} W_L^{0,0} & . & W_L^{L/2,0}. & . & W_L^{L-1,0} \\ W_L^{0,1} & . & W_L^{L/2,1}. & . & W_L^{L-1,1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{0,L-1} & . & W_L^{L/2,L-1} & . & W_L^{L-1,L-1} \end{bmatrix} \begin{bmatrix} S_0 \\ . \\ S_{L/2} \\ . \\ S_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

- The matrix to the left is the inverse Fourier matrix
- Multiplying the Fourier transform by this matrix gives us the signal right back from its Fourier transform

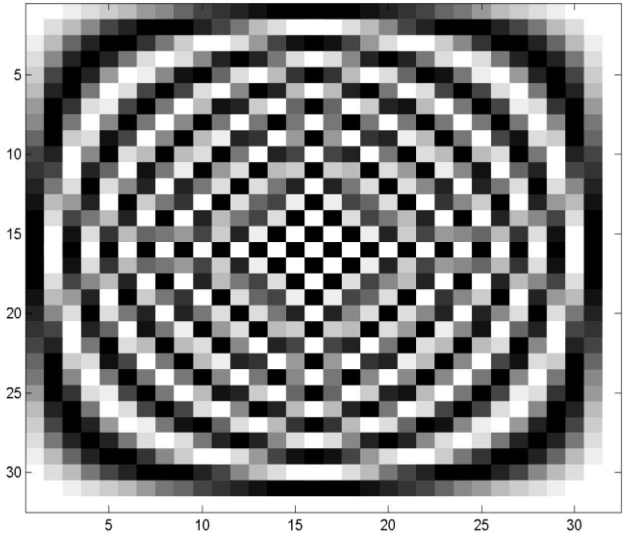
The Fourier Matrix

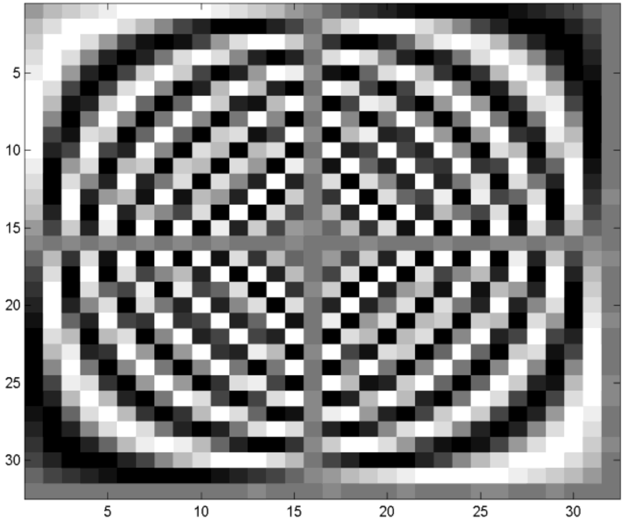




- Left panel: The real part of the Fourier matrix
  - For a 32-point signal
- Right panel: The imaginary part of the Fourier matrix

The FAST Fourier Transform



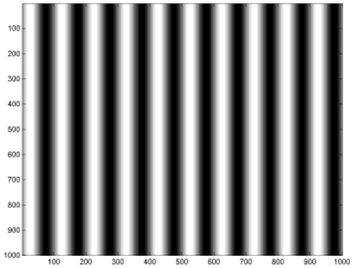


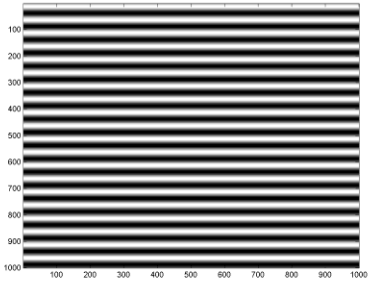
- The outcome of the transformation with the Fourier matrix is the **DISCRETE FOURIER TRANSFORM** (DFT)
- The **FAST Fourier transform** is an algorithm that takes advantage of the symmetry of the matrix to perform the matrix multiplication really fast
- The FFT computes the DFT
  - Is much faster if the length of the signal can be expressed as 2^N

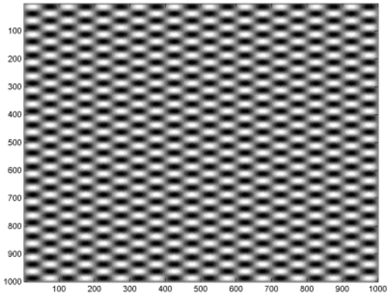
Images

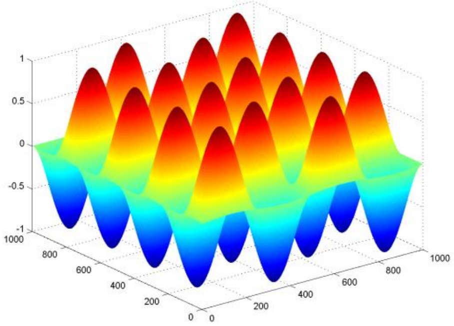
- The complex exponential is two dimensional
  - Has a separate X frequency and Y frequency
    - Would be true even for checker boards!
  - The 2-D complex exponential must be unravelled to form one component of the Fourier matrix
    - For a KxL image, we'd have K*L bases in the matrix

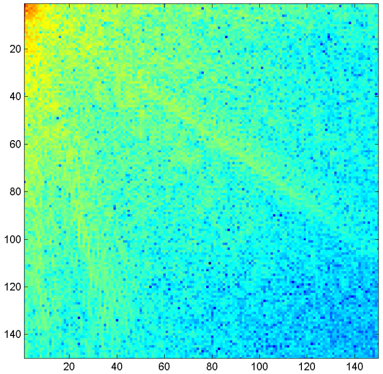
Typical Image Bases









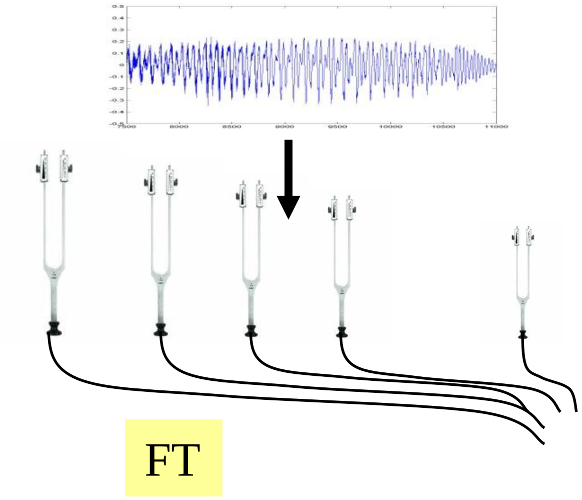


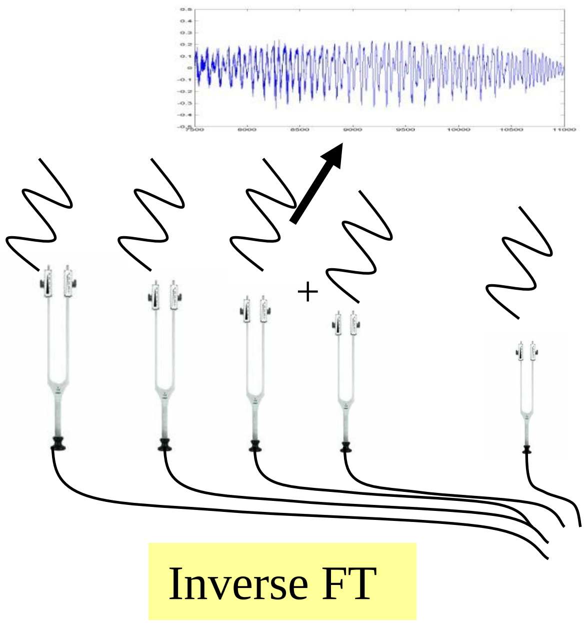
- Only real components of bases shown

The Fourier Transform and Perception: Sound

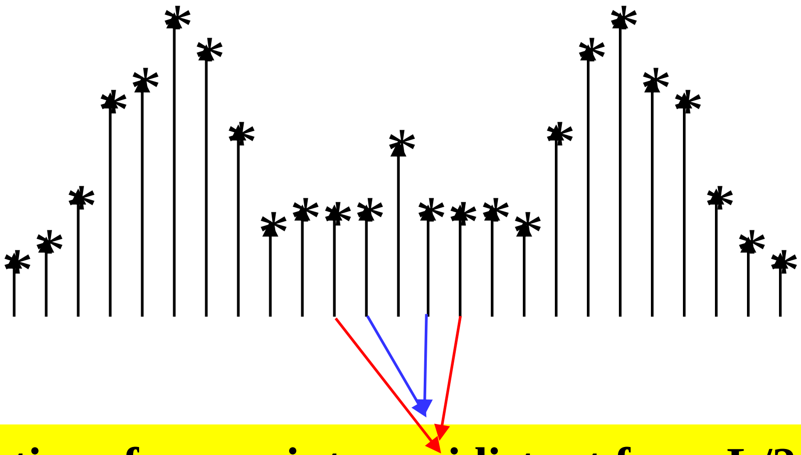
- The Fourier transforms represents the signal analogously to a bank of tuning forks

- Our ear *has* a bank of tuning forks
- The output of the Fourier transform is perceptually very meaningful





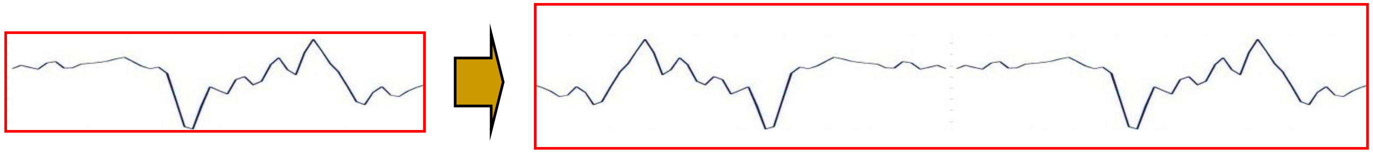
Symmetric signals

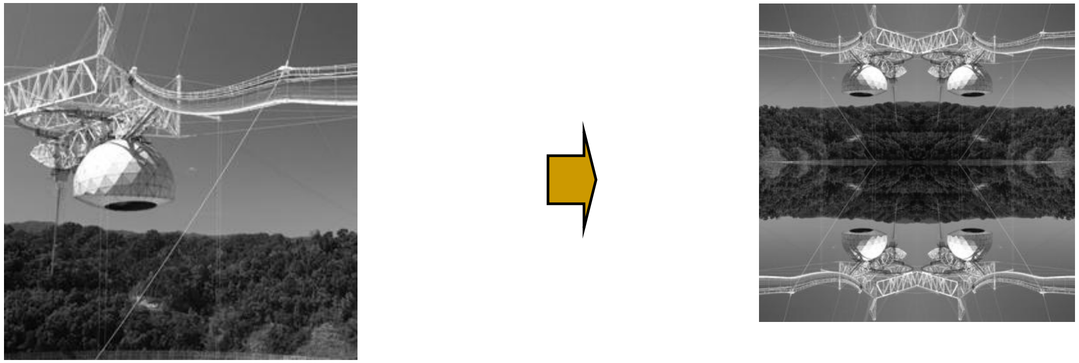


Contributions from points equidistant from L/2 combine to cancel out imaginary terms

- If a signal is (conjugate) symmetric around L/2, the Fourier coefficients are real!
  - A(L/2-k) exp(-j f(L/2-k)) + A(L/2+k) exp(-jf(L/2+k)) is always real if
    A(L/2-k) = conjugate(A(L/2+k))
  - We can pair up samples around the center all the way; the final summation term is always real
- Overall symmetry properties
  - If the *signal* is real, the FT is (conjugate) symmetric
  - If the signal is (conjugate) symmetric, the FT is real
  - **If the signal is real and symmetric, the FT is real and symmetric**

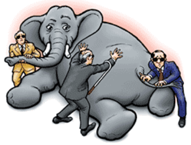
The Discrete Cosine Transform





- Compose a symmetric signal or image
  - Images would be symmetric in two dimensions

- Compute the Fourier transform
  - Since the FT is symmetric, sufficient to store only half the coefficients (quarter for an image)
    - Or as many coefficients as were originally in the signal / image



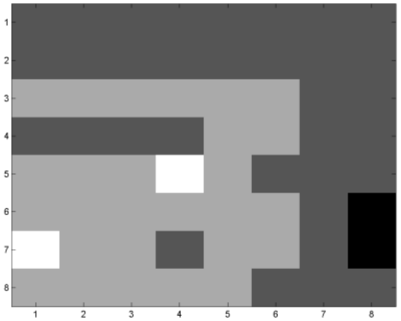
DCT

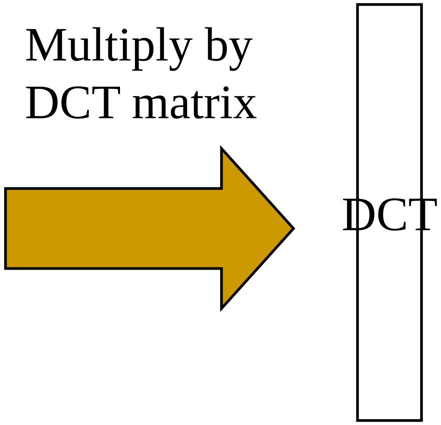
$$\begin{bmatrix} \cos(2\pi(0.5).0/2\mathrm{L}) & \cos(2\pi.(1+0.5).0/2\mathrm{L}) & . & . & \cos(2\pi.(L-0.5).0/2\mathrm{L}) \\ \cos(2\pi.(0.5).1/2\mathrm{L}) & \cos(2\pi.(1+0.5).1/2\mathrm{L}) & . & . & \cos(2\pi.(L-0.5).1/2\mathrm{L}) \\ . & . & . & . & . \\ . & . & . & . & . \\ \cos(2\pi.(0.5).(L-1)/2\mathrm{L}) & \cos(2\pi.(1+0.5).(L-1)/2\mathrm{L}) & . & . & \cos(2\pi.(L-0.5).(L-1)/2\mathrm{L}) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ w_{L-1} \end{bmatrix} = \begin{bmatrix} s[0] \\ s[1] \\ . \\ . \\ s[L-1] \end{bmatrix}$$

L columns

- Not necessary to compute a 2xL sized FFT
  - Enough to compute an L-sized *cosine* transform
  - Taking advantage of the symmetry of the problem
- This is the Discrete Cosine Transform

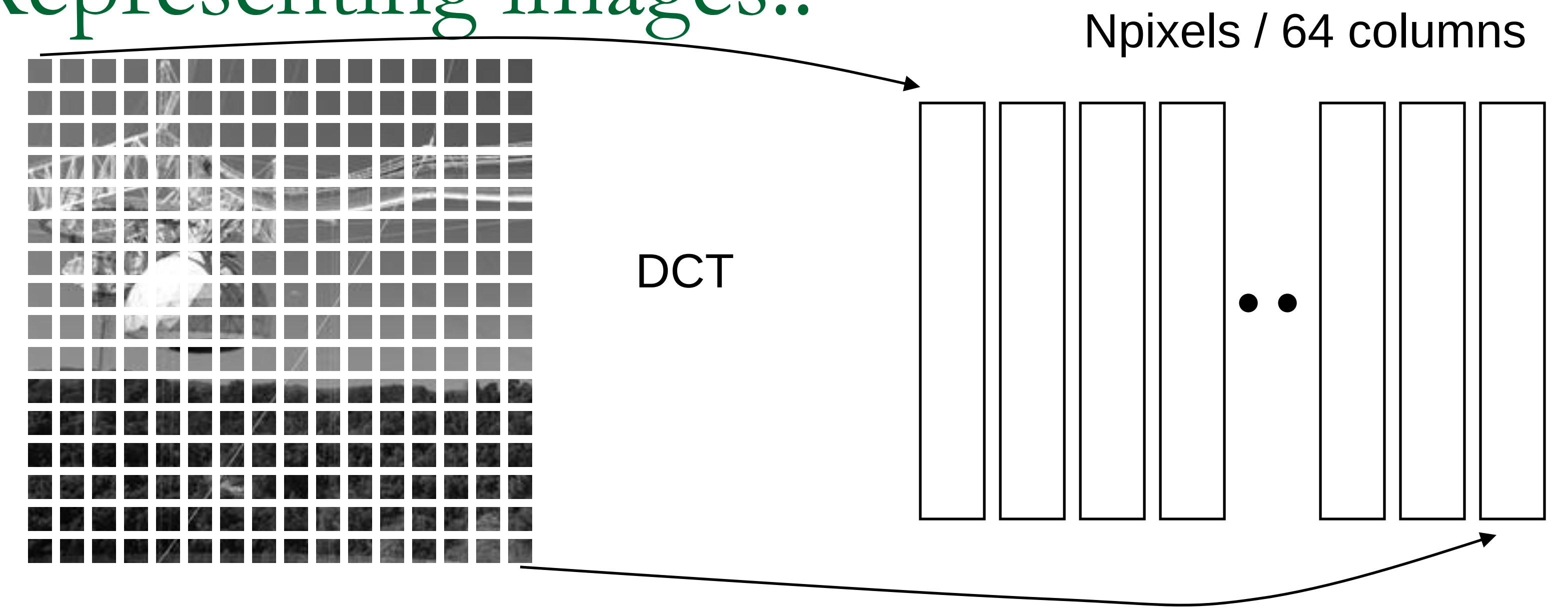
Representing images




Multiply by DCT matrix

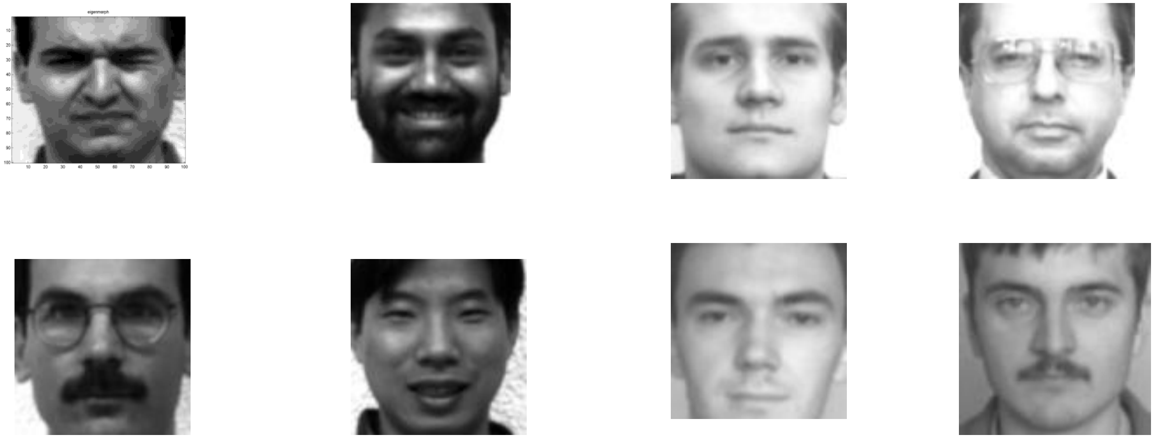
- Most common coding is the DCT
- JPEG: Each 8x8 element of the picture is converted using a DCT
- The DCT coefficients are quantized and stored
  - Degree of quantization = degree of compression
- Also used to represent textures etc for pattern recognition and other forms of analysis

Representing images..



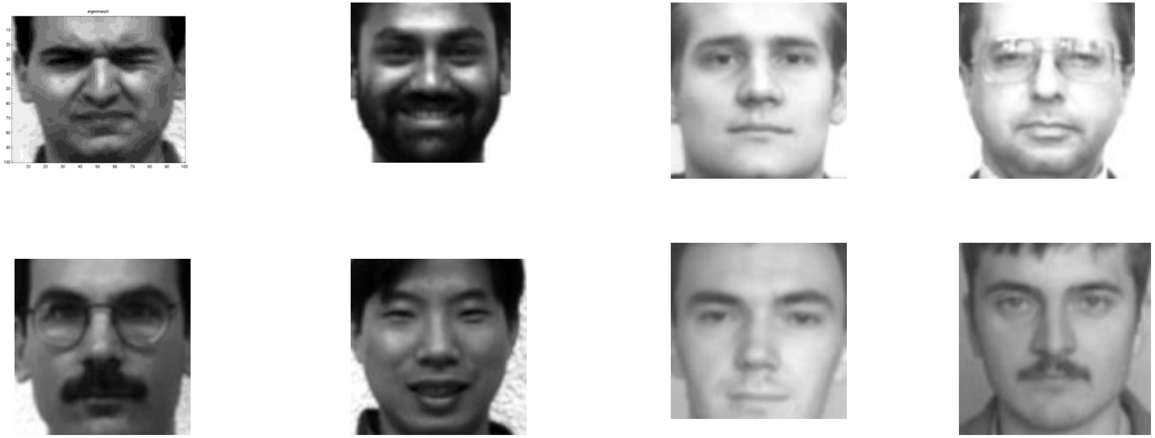
- DCT of small segments
  - 8x8
  - Each image becomes a matrix of DCT vectors
- DCT of the image
- This is a *data agnostic transform representation*
- ***Or data-driven representations..***

Returning to Eigen Computation

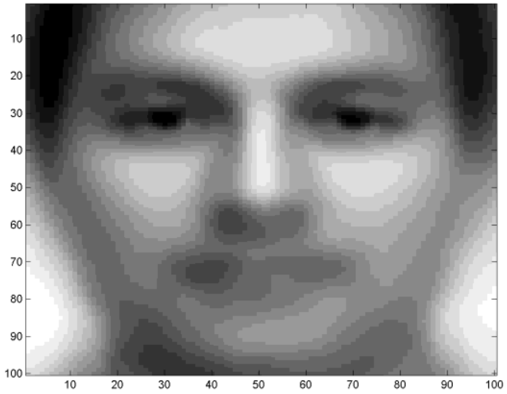


- A collection of faces
  - All normalized to 100x100 pixels
- What is common among all of them?
  - Do we have a common descriptor?

A least squares typical face

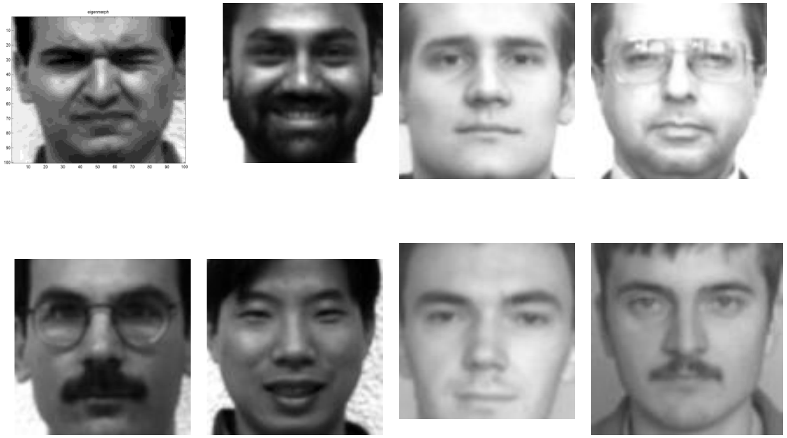


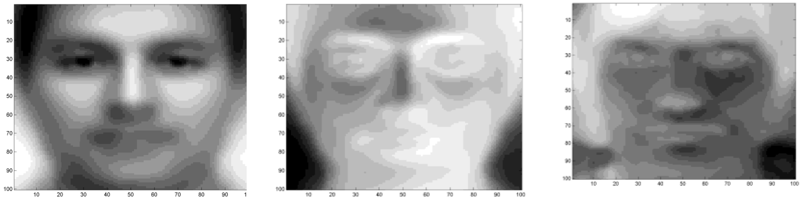
The typical face



- Can we do better than a blank screen to find the most common portion of faces?
  - The first checkerboard; the zeroth frequency component..
- Assumption: There is a “typical” face that captures most of what is common to all faces
  - Every face can be represented by a scaled version of a typical face
    - What is this face?
- Approximate **every** face f as $f = w_f V$
- Estimate V to minimize the squared error
  - How?
  - What is V?

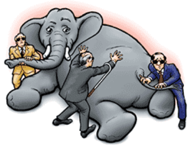
A collection of least squares typical faces





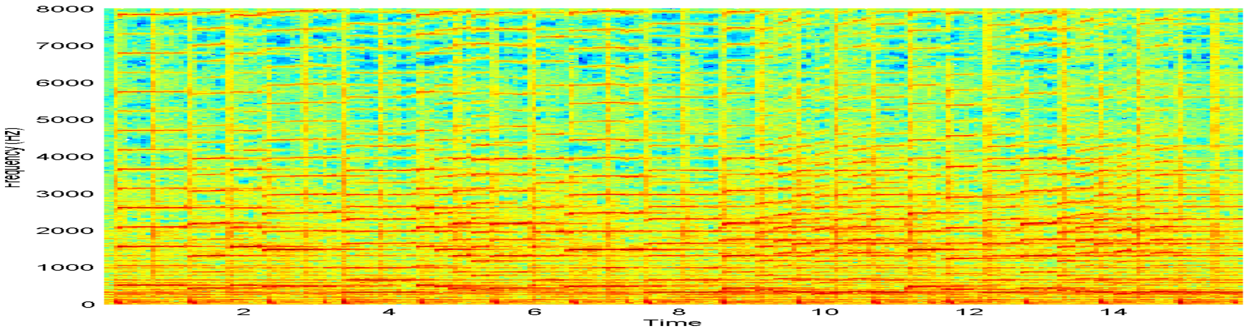
..

- Assumption: There are a set of K "typical" faces that captures most of all faces
- Approximate **every** face f as $f = w_{f,1} V_1 + w_{f,2} V_2 + w_{f,3} V_3 +.. + w_{f,k} V_k$
  - V_2 is used to "correct" errors resulting from using only V_1
    - So the total energy in $w_{f,2}$ ($\Sigma\, w_{f,2}^2$) must be lesser than the total energy in $w_{f,1}$ ($\Sigma\, w_{f,1}^2$)
  - V_3 corrects errors remaining after correction with V_2
    - The total energy in $w_{f,3}$ must be lesser than that even in $w_{f,2}$
  - And so on..
  - $V = [V_1\ V_2\ V_3]$
- Estimate V to minimize the squared error
  - How?
  - What is V?



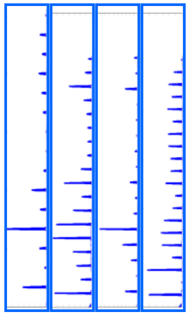
A recollection

M =



V=PINV(W)*M

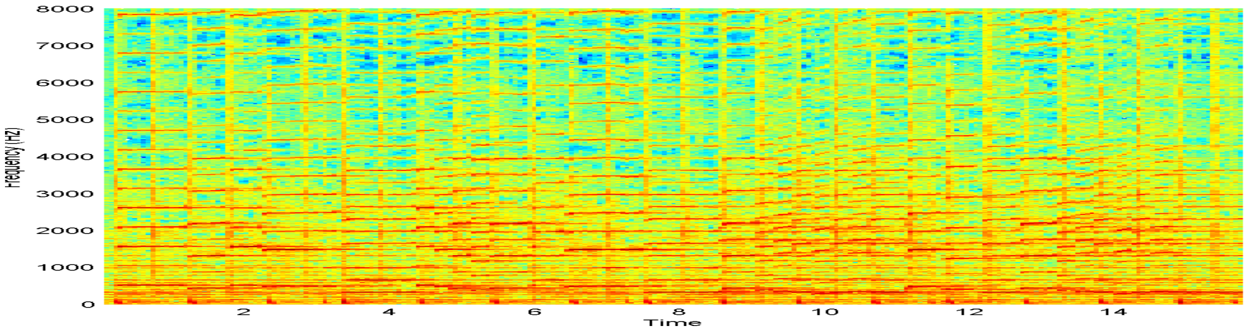
W =



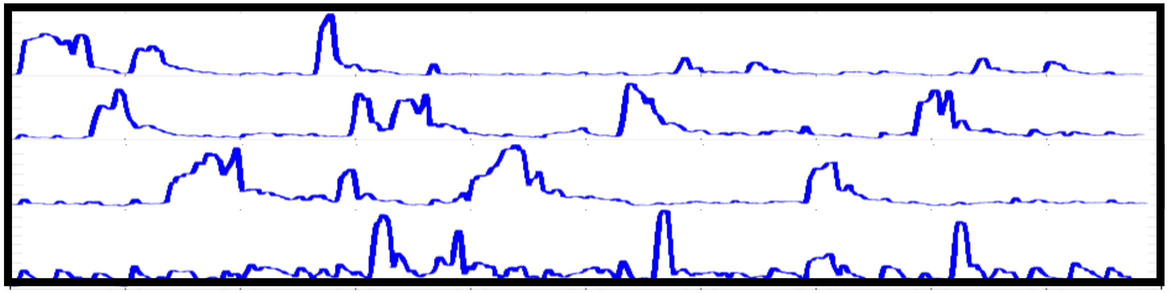
U = ?

How about the other way?

M =



V =

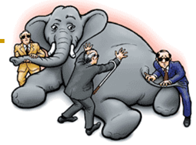


W = ?

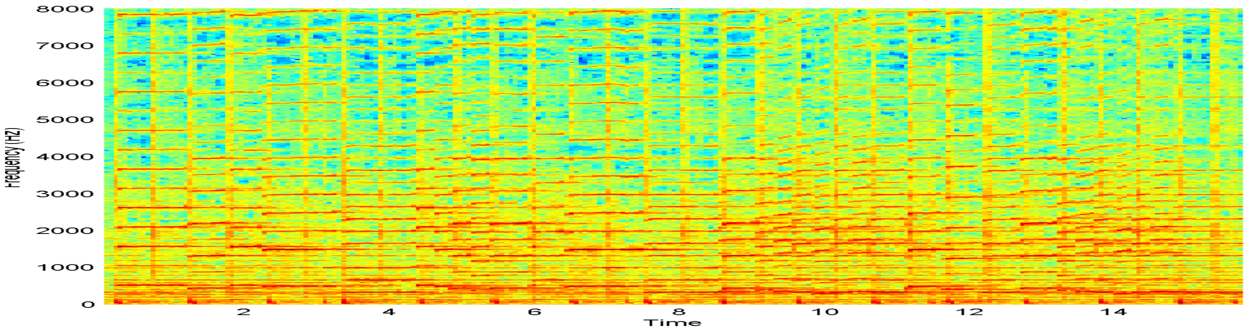
U = ?

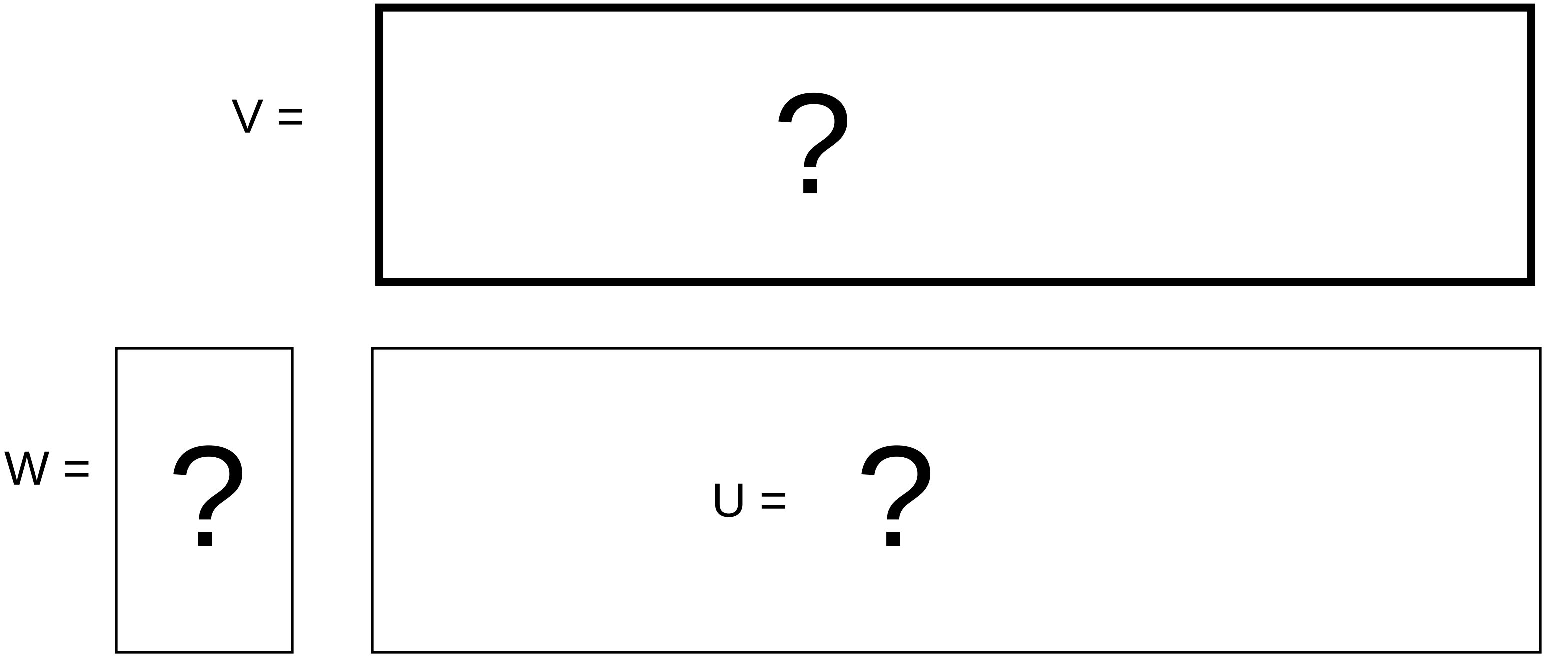
- W = M * Pinv(V)

How about the other way?



M =





V =

?

W =

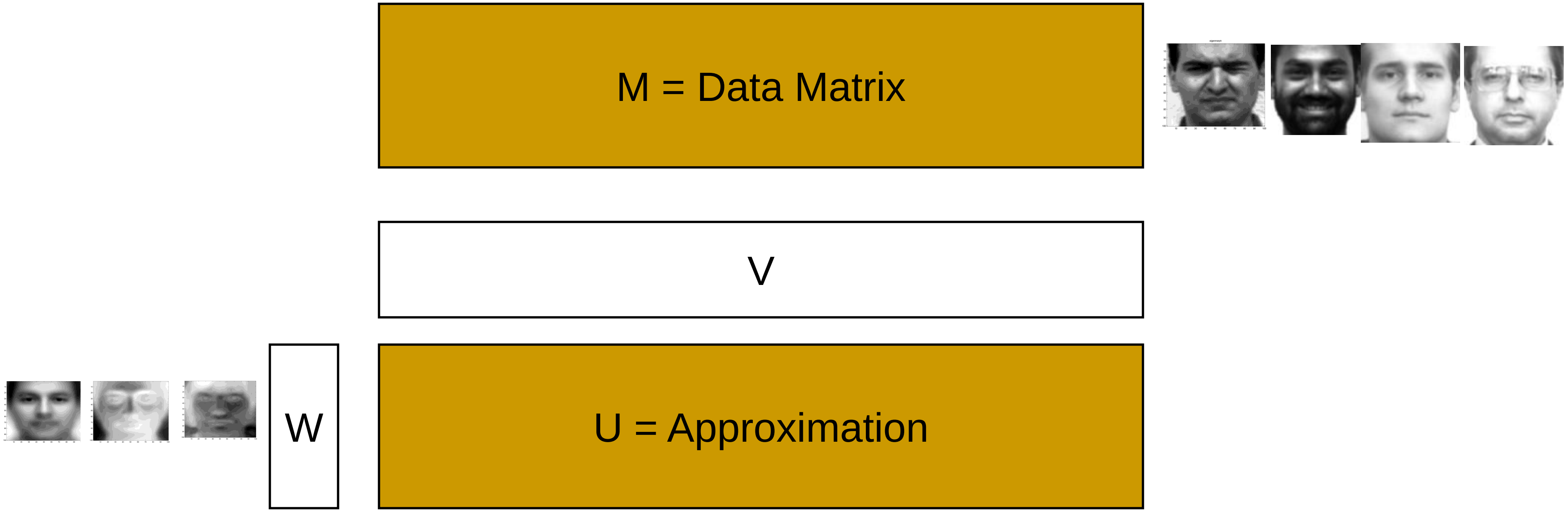
?

U =

?

- W V \approx = M

Eigen Faces!


M = Data Matrix

V

W

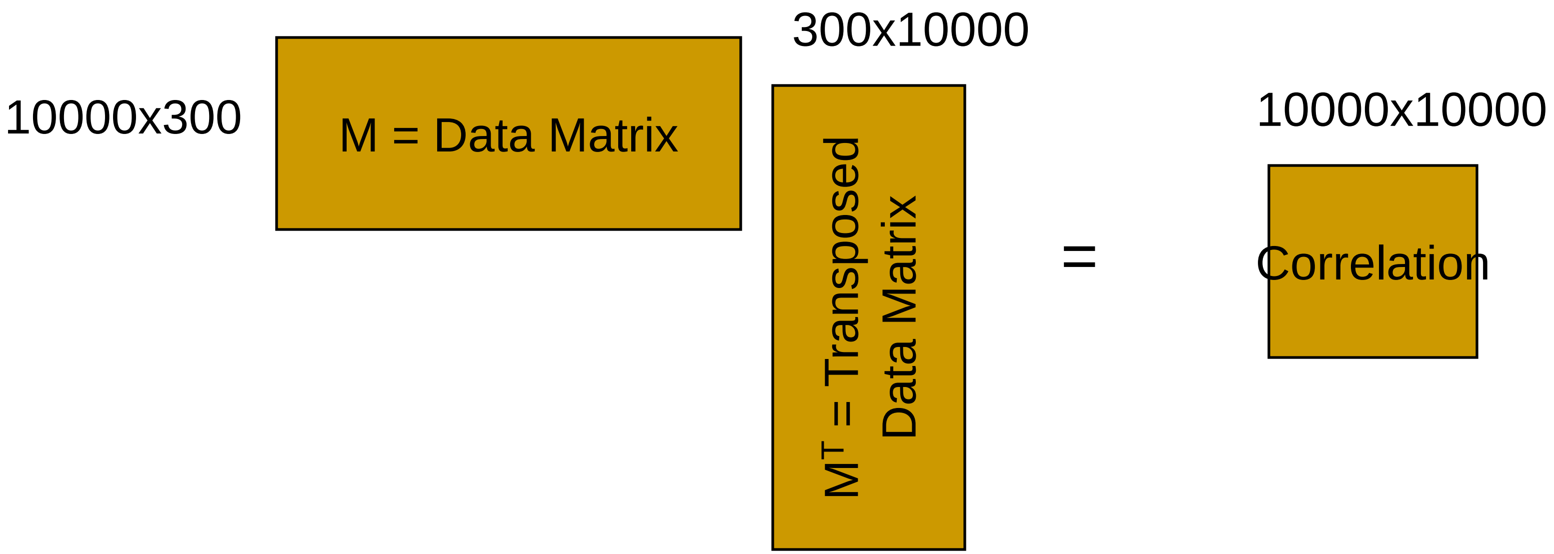
U = Approximation

- Here W, V and U are ALL unknown and must be determined
    - Such that the squared error between U and M is minimum

- Eigen analysis allows you to find W and V such that U = WV has the least squared error with respect to the original data M

- If the original data are a collection of faces, the columns of W represent the space of *eigen faces.*

Eigen faces



10000x300

M = Data Matrix

300x10000

M^T = Transposed Data Matrix

=

10000x10000

Correlation

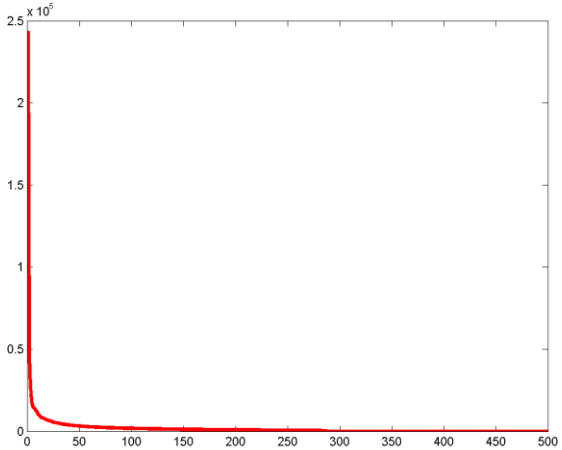
- Lay all faces side by side in vector form to form a matrix
  - In my example: 300 faces. So the matrix is 10000 x 300
- Multiply the matrix by its transpose
  - The correlation matrix is 10000x10000

Eigen faces

```
[U,S] = eig(correlation)
```

$$S = \begin{bmatrix} \lambda_1 & . & 0 & . & 0 \\ 0 & \lambda_2 & 0 & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & . & 0 & . & \lambda_{10000} \end{bmatrix}$$

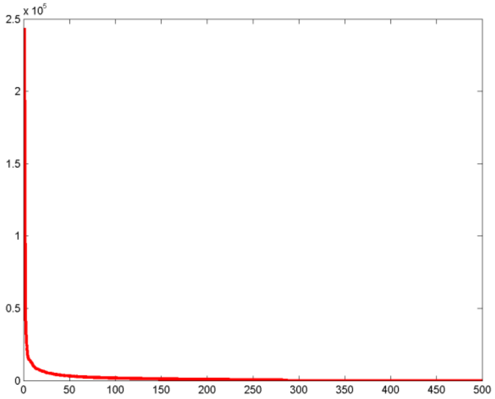
$$U = \begin{bmatrix} \text{eigenface1} & \text{eigenface2} & \bullet\bullet\bullet \end{bmatrix}$$



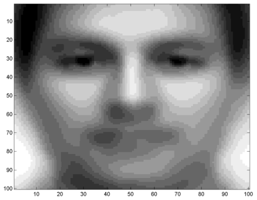
- Compute the eigen vectors
  - Only 300 of the 10000 eigen values are non-zero
    - Why?
- Retain eigen vectors with high eigen values (>0)
  - Could use a higher threshold

Eigen Faces

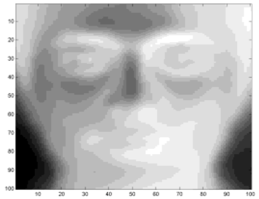
$$U = \begin{bmatrix} \text{eigenface1} & \text{eigenface2} & \bullet\bullet\bullet \end{bmatrix}$$

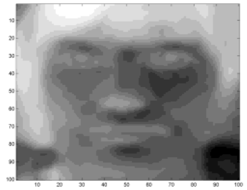


eigenface1



eigenface2



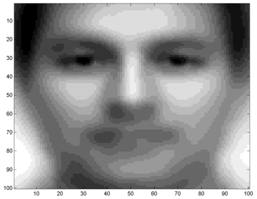


eigenface3

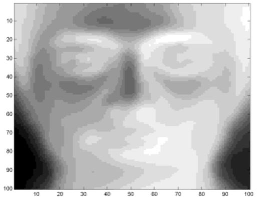
- The eigen vector with the highest eigen value is the first typical face
- The vector with the second highest eigen value is the second typical face.
- Etc.

Representing a face

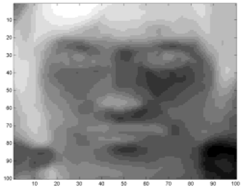
= w_1



+ w_2

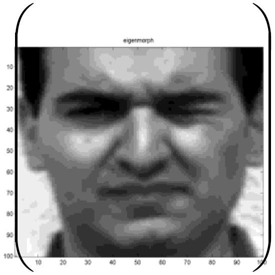


+ w_3



•••

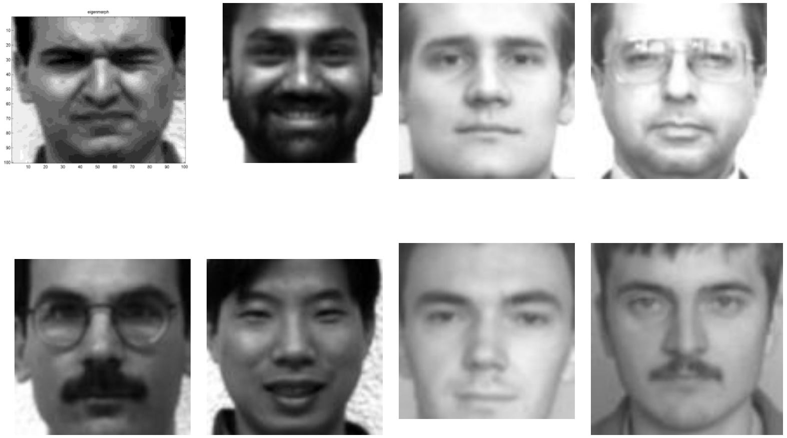
Representation (

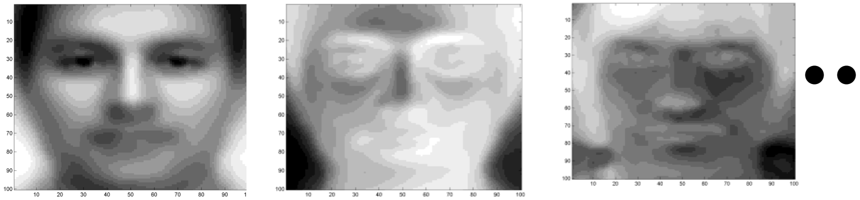


) = $[w_1\ w_2\ w_3\ ....\ ]^T$

- The weights with which the eigen faces must be combined to compose the face are used to represent the face!

The Energy Compaction Property

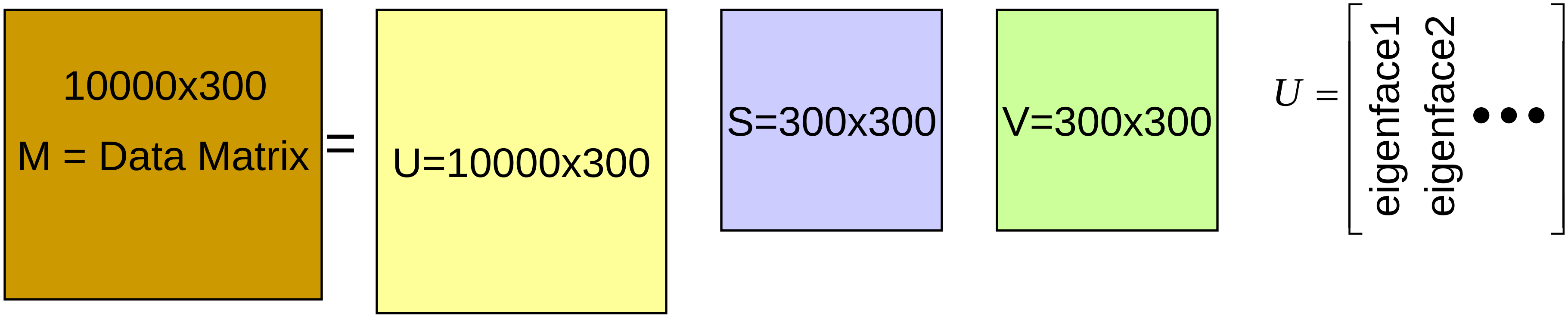




- The first K Eigen faces (for any K) represent the *best possible way to represent the data*
    - In an L2 sense

- $\Sigma_f \Sigma_k w_{f,k}^2$ cannot be lesser for an other set of “typical” faces
    - Almost by definition
    - This was the requirement posed in our “least squares” estimation.

SVD instead of Eigen



10000x300

M = Data Matrix

=

U=10000x300

S=300x300

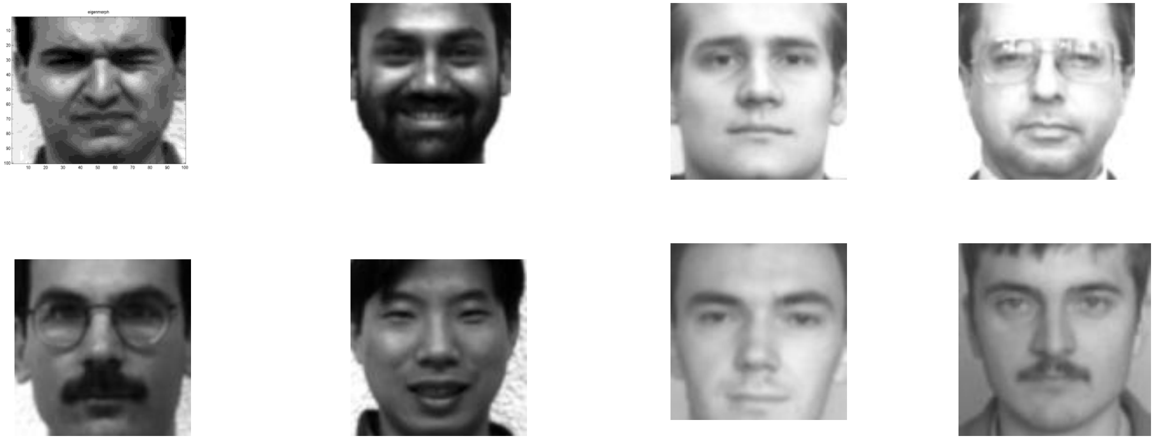
V=300x300

$U = \begin{bmatrix} \text{eigenface1} & \text{eigenface2} & \cdots \end{bmatrix}$

- Do we need to compute a 10000 x 10000 correlation matrix and then perform Eigen analysis?
  - Will take a very long time on your laptop
- SVD
  - Only need to perform “Thin” SVD. Very fast
    - U = 10000 x 300
      - The columns of U are the eigen faces!
      - The Us corresponding to the “zero” eigen values are not computed
    - S = 300 x 300
    - V = 300 x 300

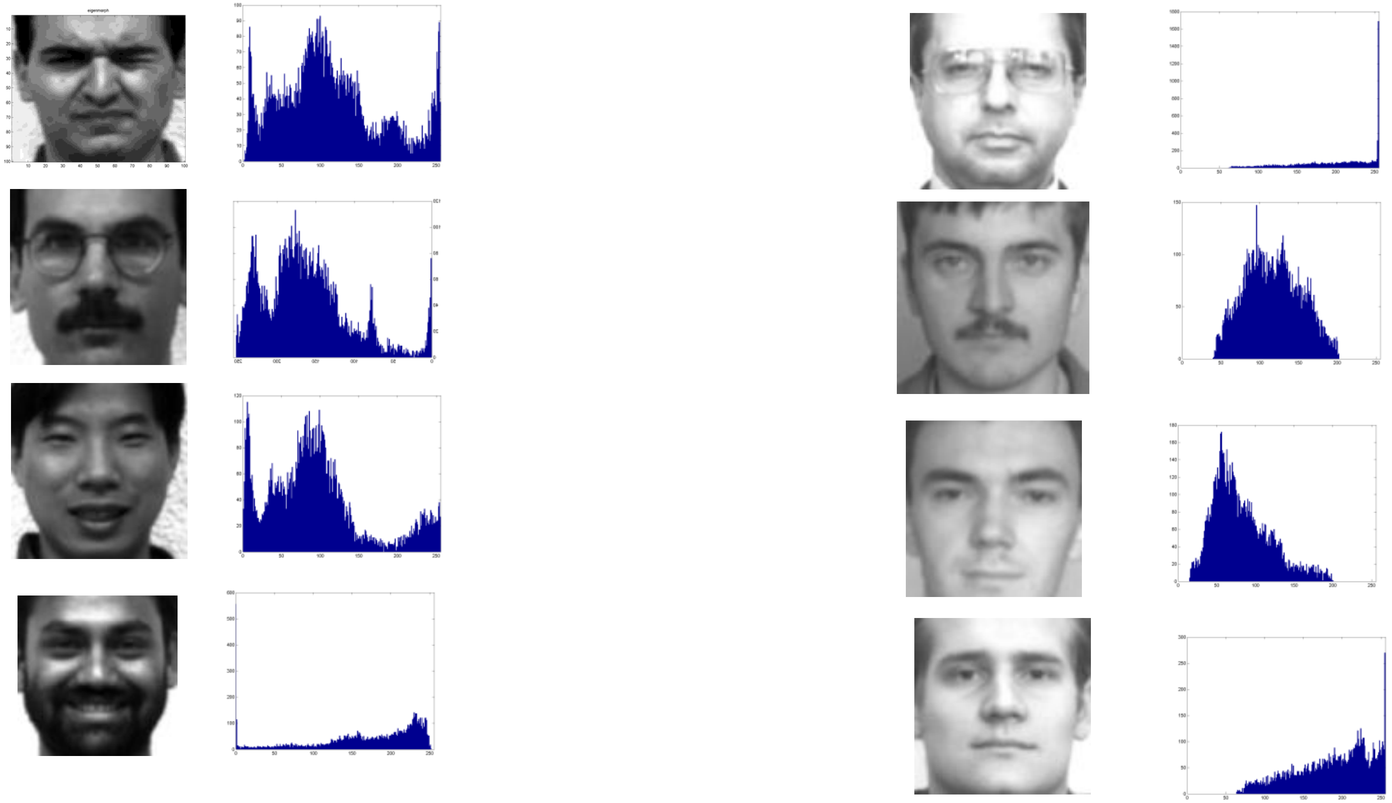
NORMALIZING OUT VARIATIONS

Images: Accounting for variations



- What are the obvious differences in the above images
- How can we capture these differences
  - Hint – image histograms..

Images -- Variations



- Pixel histograms: what are the differences

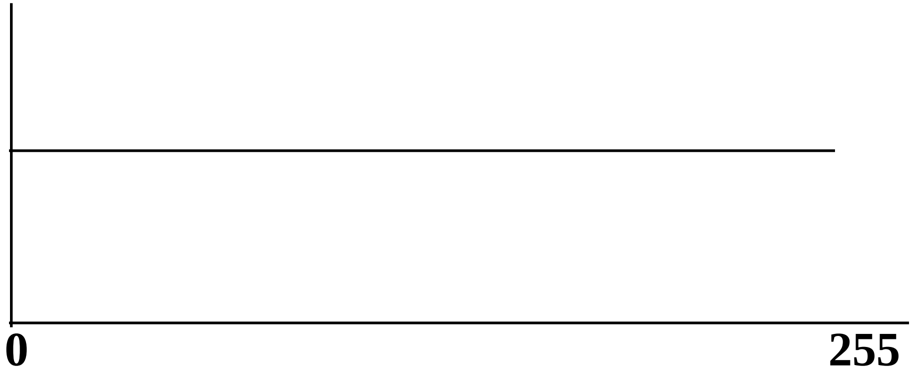
Normalizing Image Characteristics

- Normalize the pictures
  - Eliminate lighting/contrast variations
  - All pictures must have “similar” lighting
    - How?

- Lighting and contrast are represented in the image histograms:

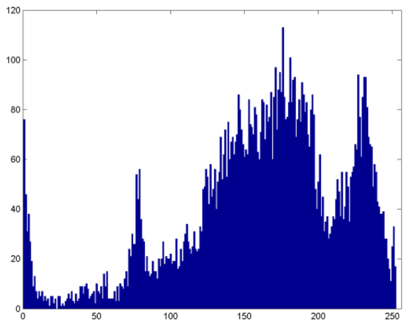
Histogram Equalization

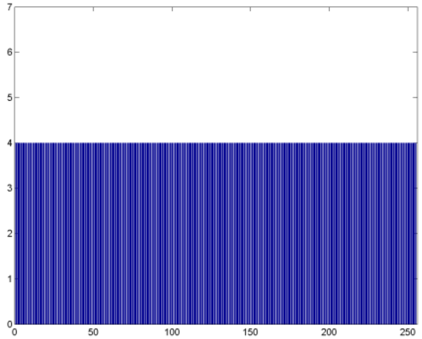
- Normalize histograms of images
  - Maximize the contrast
    - Contrast is defined as the “flatness” of the histogram
    - For maximal contrast, every greyscale must happen as frequently as every other greyscale



- Maximizing the contrast: Flattening the histogram
  - Doing it for every image ensures that every image has the same constrast
    - I.e. exactly the same histogram of pixel values
      - Which should be flat

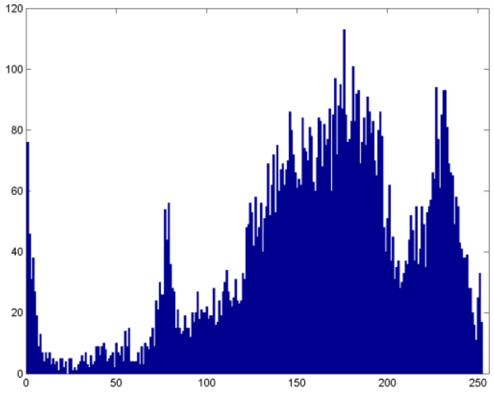
Histogram Equalization

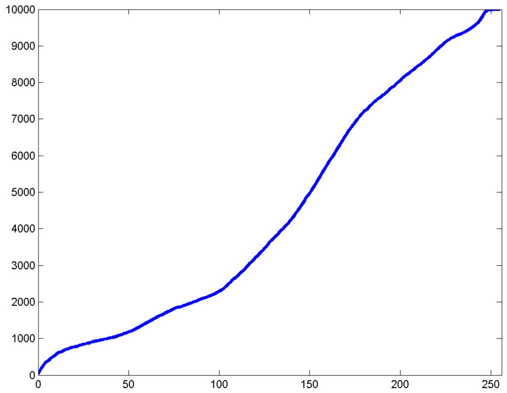




- Modify pixel values such that histogram becomes “flat”.
- For each pixel
  - New pixel value = f(old pixel value)
  - What is f()?
- Easy way to compute this function: map cumulative counts

Cumulative Count Function

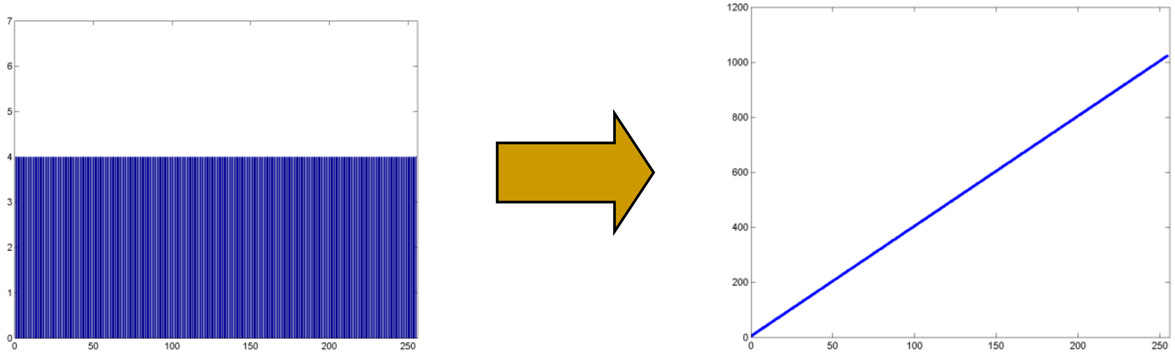




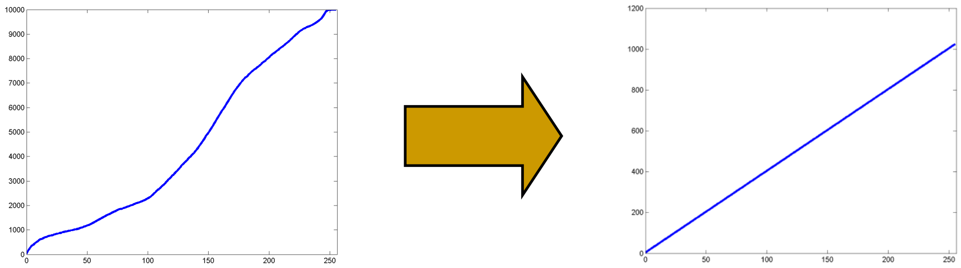
- The *histogram (count)* of a pixel value X is the number of pixels in the image that have value X
  - E.g. in the above image, the count of pixel value 180 is about 110

- The *cumulative count* at pixel value X is the total number of pixels that have values in the range 0 <= x <= X
  - CCF(X) = H(1) + H(2) + .. H(X)

Cumulative Count Function

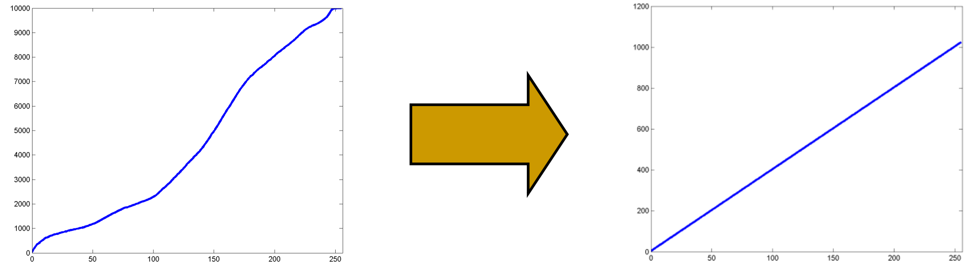


- The cumulative count function of a uniform histogram is a line



- We must modify the pixel values of the image so that its cumulative count is a line

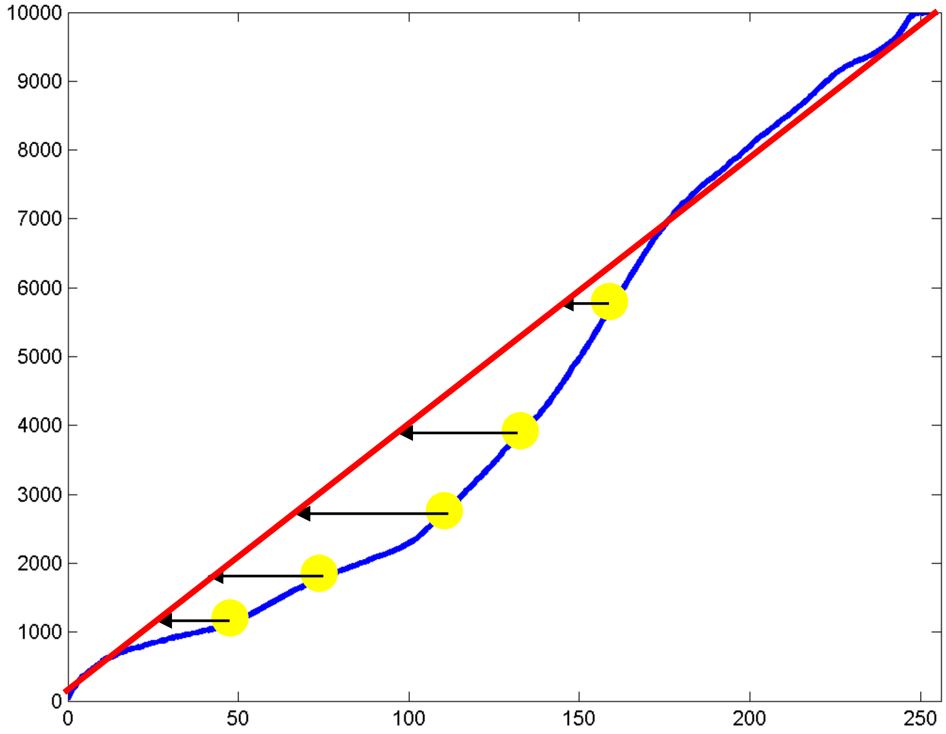
Mapping CCFs



Move x axis levels around until the plot to the left looks like the plot to the right

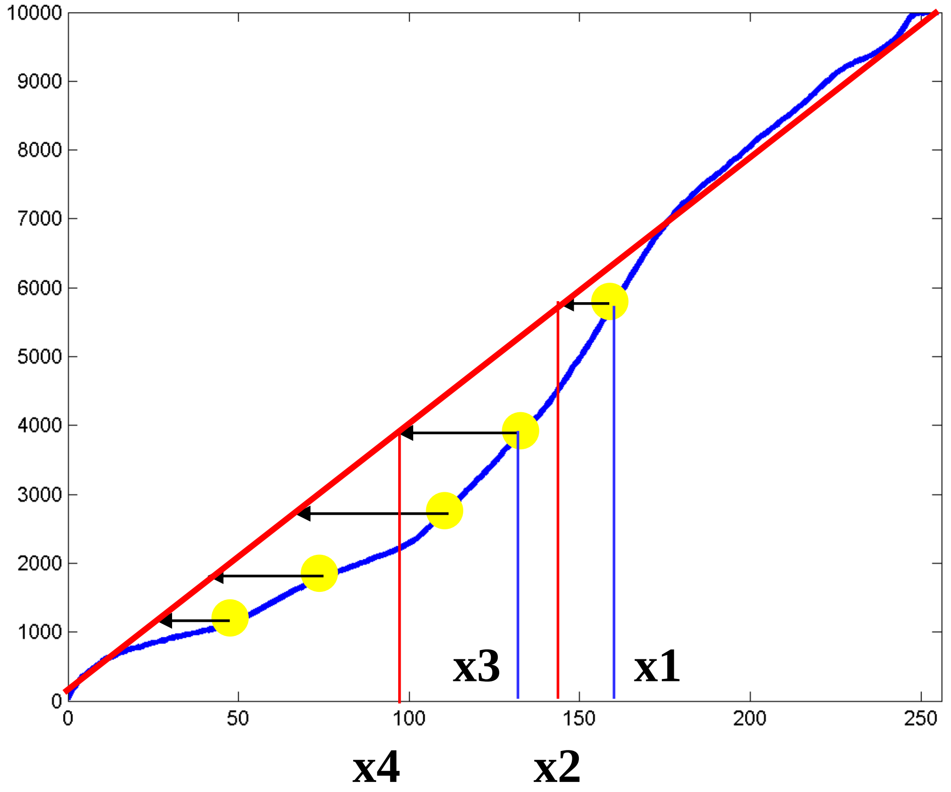
- CCF(f(x)) -> a*f(x)   [or a*(f(x)+1) if pixels can take value 0]
  - x = pixel value
  - f() is the function that converts the old pixel value to a new (normalized) pixel value
  - a = (total no. of pixels in image) / (total no. of pixel levels)
    - The no. of pixel levels is 256 in our examples
    - Total no. of pixels is 10000 in a 100x100 image

Mapping CCFs



- For each pixel value x:
  - Find the location on the red line that has the closet Y value to the observed CCF at x

Mapping CCFs



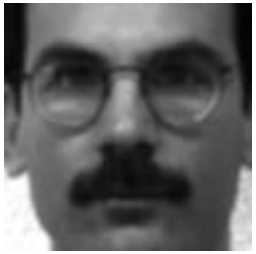
f(x1) = x2

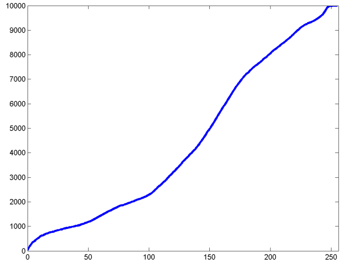
f(x3) = x4

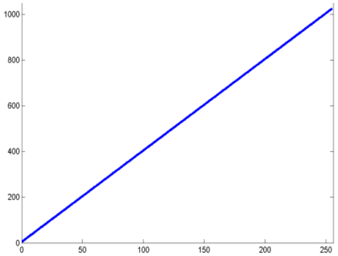
Etc.

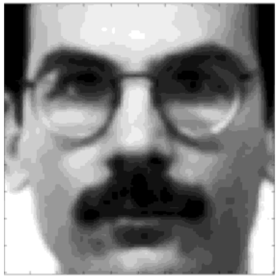
- For each pixel value x:
  - Find the location on the red line that has the closet Y value to the observed CCF at x

Mapping CCFs





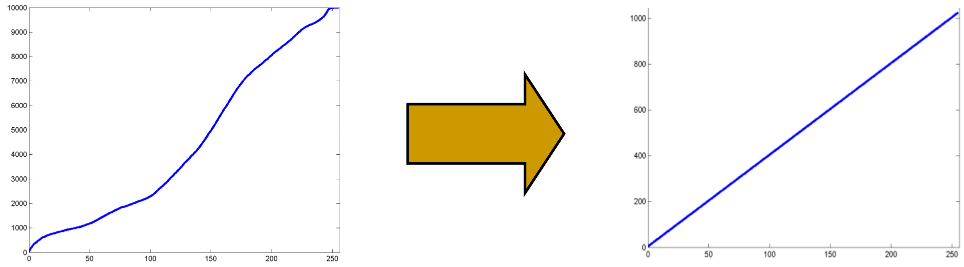




Move x axis levels around until the plot to the left looks like the plot to the right

- For each pixel in the image to the left
  - The pixel has a value x
  - Find the CCF at that pixel value CCF(x)
  - Find x’ such that CCF(x’) in the function to the right equals CCF(x)
    - x’ such that CCF_flat(x’) = CCF(x)
  - Modify the pixel value to x’

Doing it Formulaically



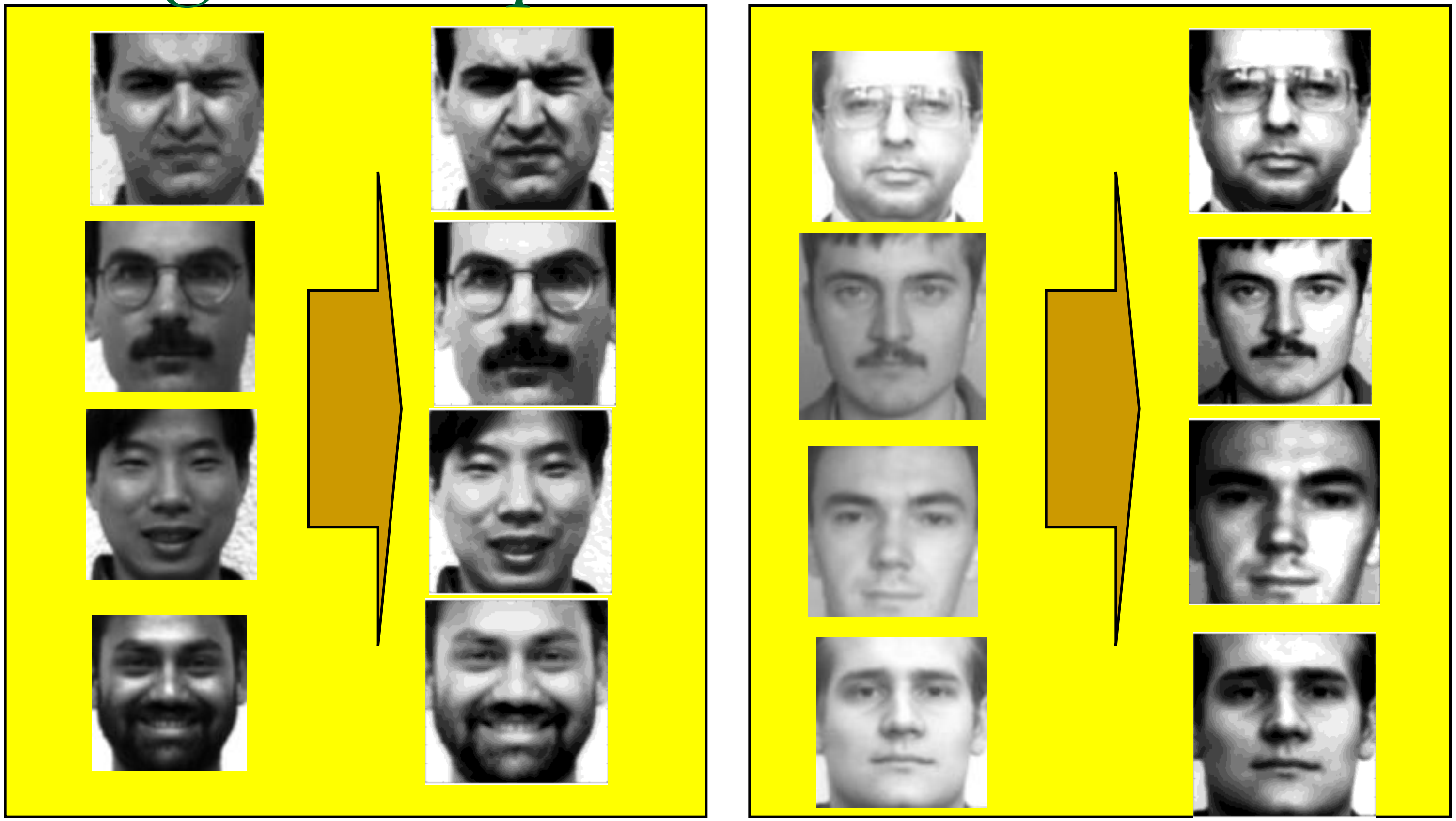
$$f(x) = round\left(\frac{CCF(x) - CCF_{\min}}{Npixels - CCF_{\min}} Max.pixel.value \right)$$

- CCF_{min} is the smallest non-zero value of CCF(x)
  - The value of the CCF at the smallest observed pixel value
- Npixels is the total no. of pixels in the image
  - 10000 for a 100x100 image
- Max.pixel.value is the highest pixel value
  - 255 for 8-bit pixel representations

Or even simpler

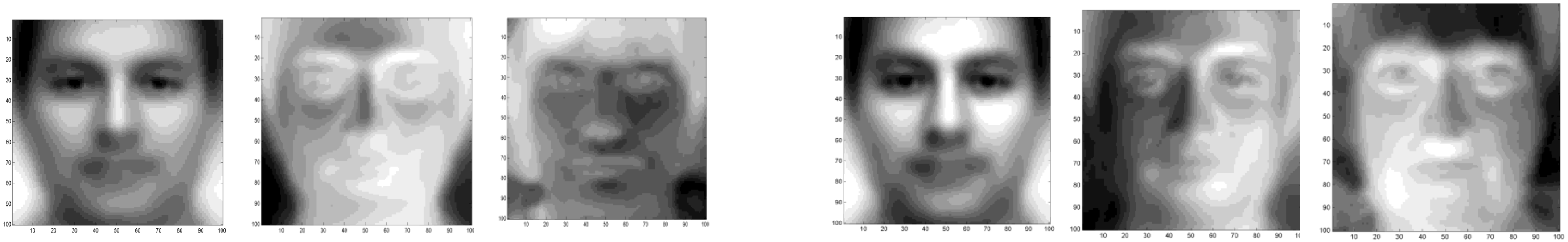
- Matlab:
  - `Newimage = histeq(oldimage)`

Histogram Equalization



- Left column: Original image
- Right column: Equalized image
- All images now have similar contrast levels

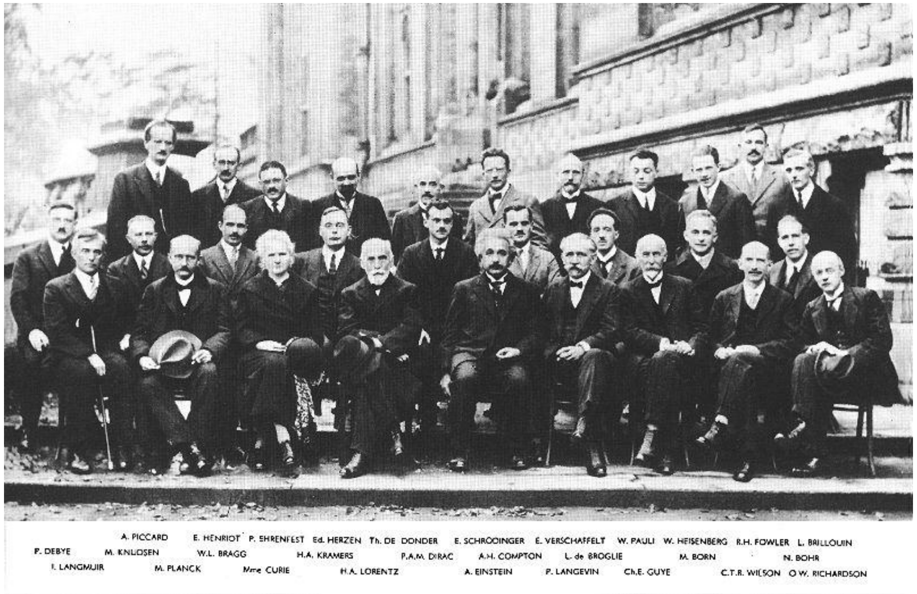
Eigenfaces after Equalization



- Left panel : Without HEQ
- Right panel: With HEQ
  - Eigen faces are more face like..
    - Need not always be the case

Detecting Faces in Images

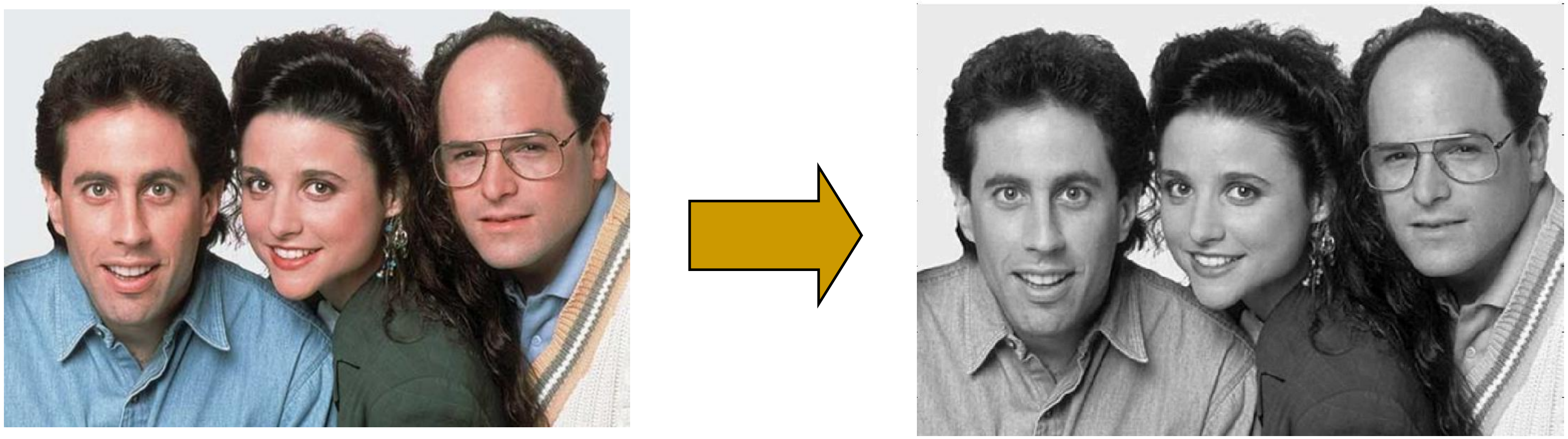
Detecting Faces in Images



- Finding face like patterns
  - How do we find if a picture has faces in it
  - Where are the faces?

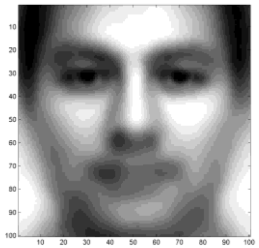
- A simple solution:
  - Define a “typical face”
  - Find the “typical face” in the image

Finding faces in an image



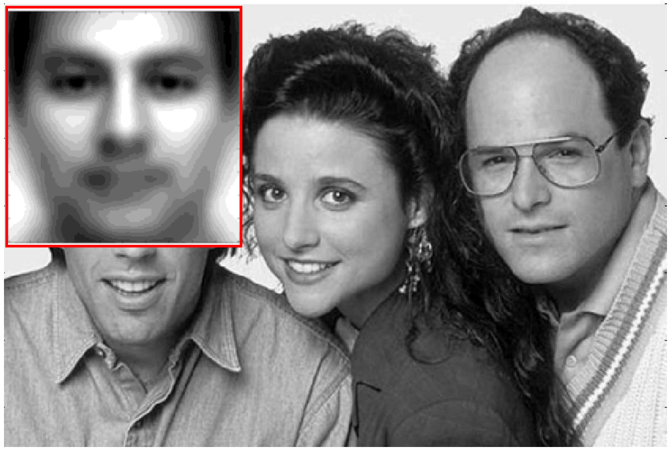
- Picture is larger than the “typical face”
  - E.g. typical face is 100x100, picture is 600x800
- First convert to greyscale
  - R + G + B
  - Not very useful to work in color

Finding faces in an image



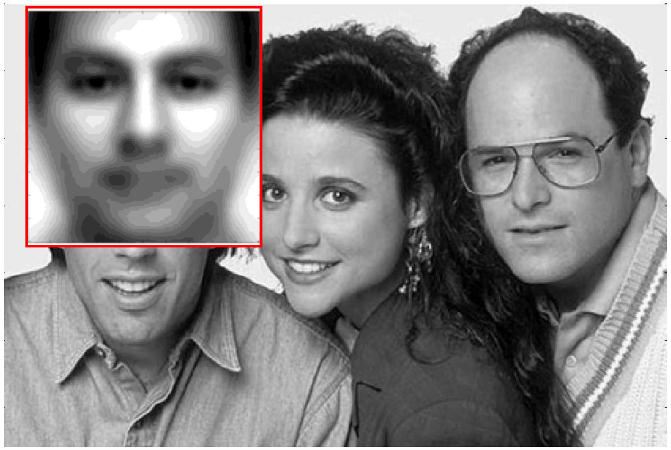
- Goal .. To find out if and where images that look like the “typical” face occur in the picture

Finding faces in an image



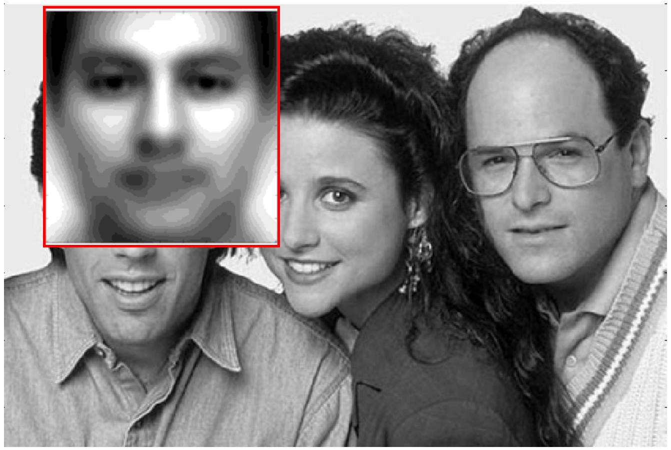
- Try to "match" the typical face to each location in the picture

Finding faces in an image



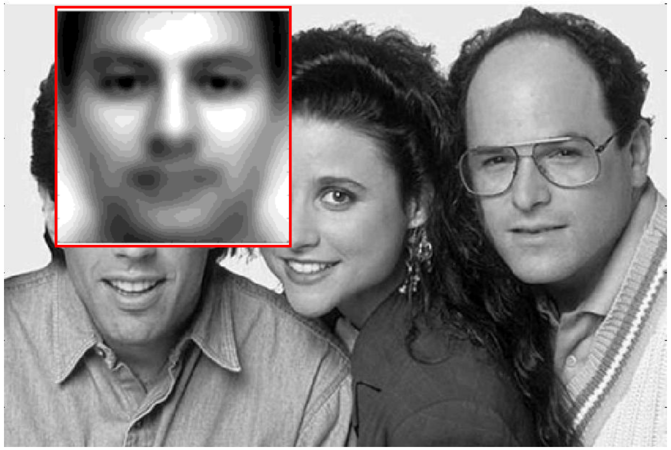
- Try to “match” the typical face to each location in the picture

Finding faces in an image



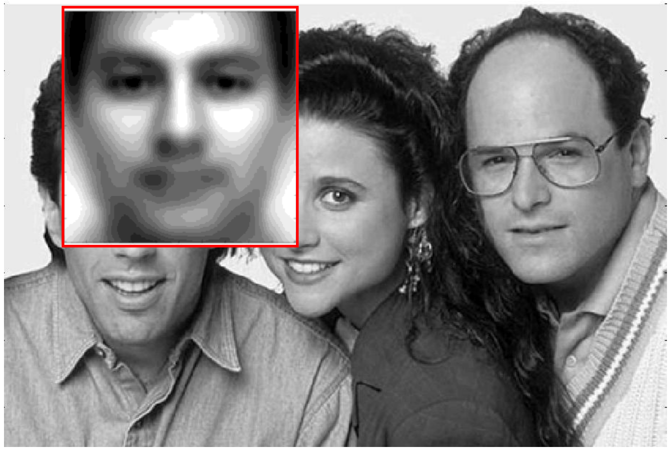
- Try to “match” the typical face to each location in the picture

Finding faces in an image



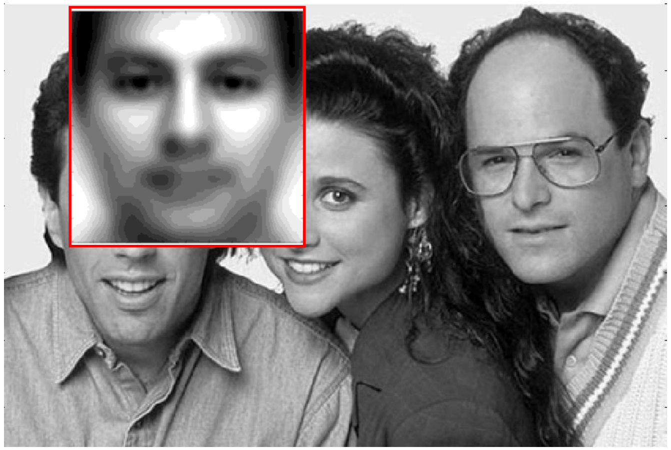
- Try to “match” the typical face to each location in the picture

Finding faces in an image



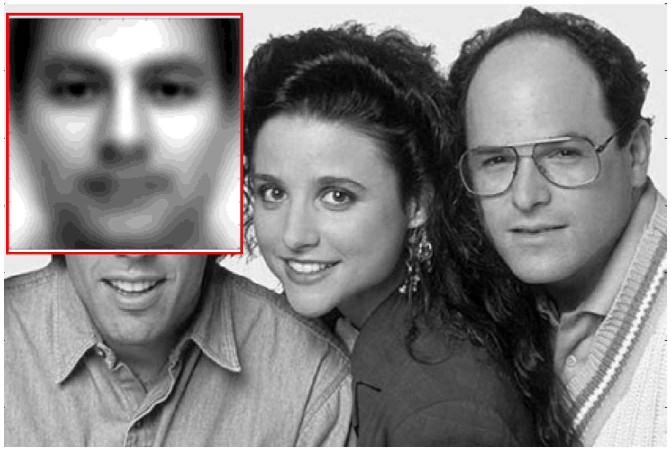
- Try to “match” the typical face to each location in the picture

Finding faces in an image



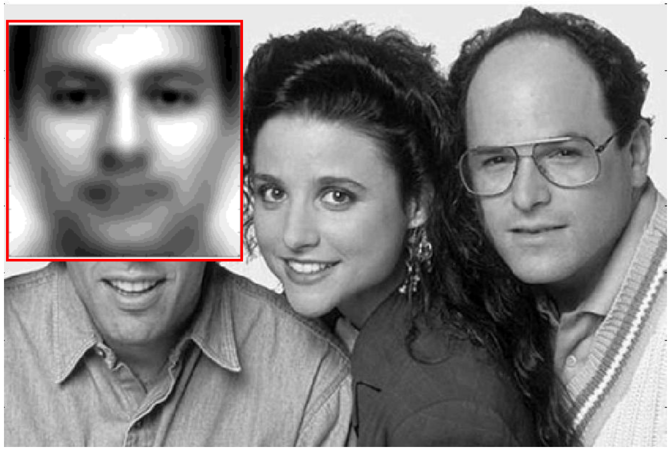
- Try to “match” the typical face to each location in the picture

Finding faces in an image



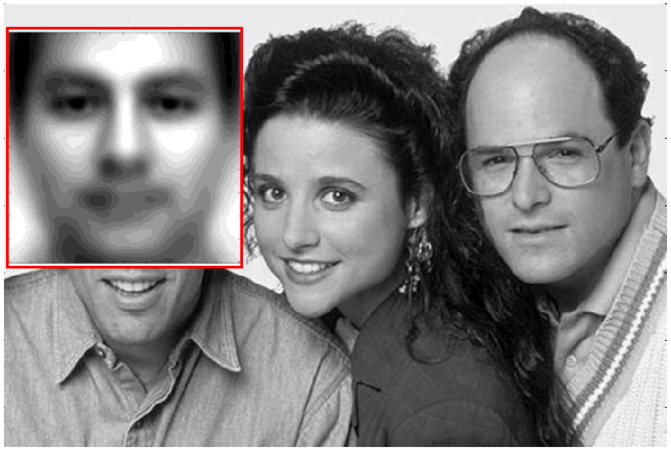
- Try to “match” the typical face to each location in the picture

Finding faces in an image



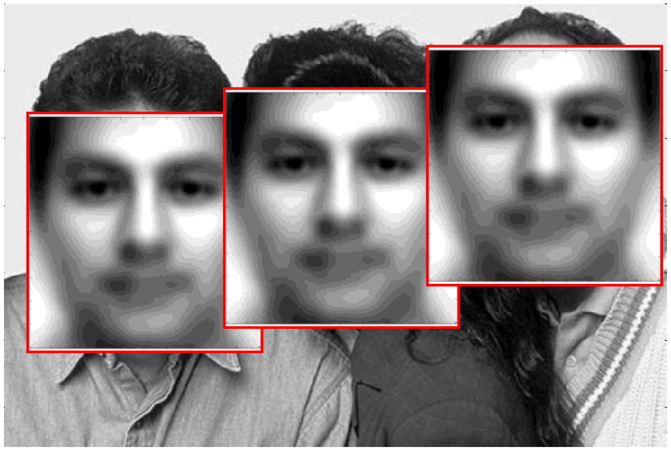
- Try to “match” the typical face to each location in the picture

Finding faces in an image



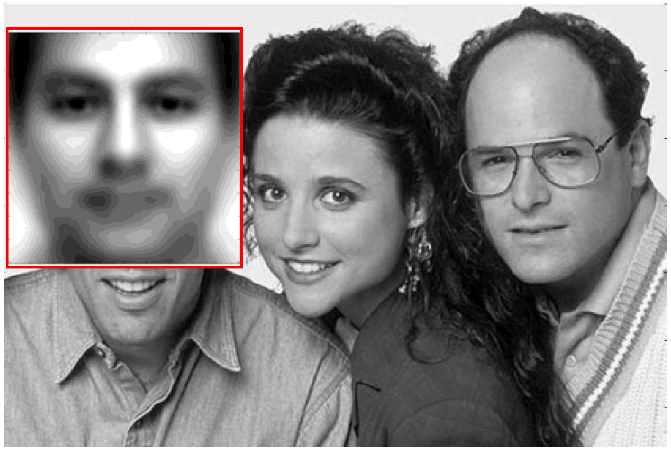
- Try to “match” the typical face to each location in the picture

Finding faces in an image



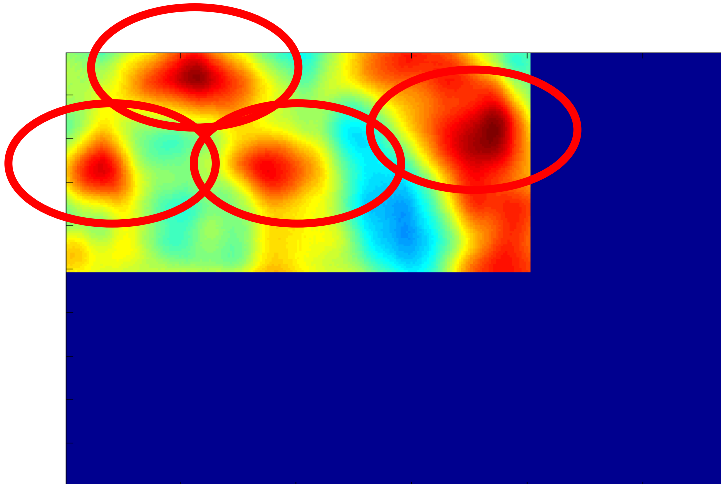
- Try to “match” the typical face to each location in the picture
- The “typical face” will explain some spots on the image much better than others
  - These are the spots at which we probably have a face!

How to "match"



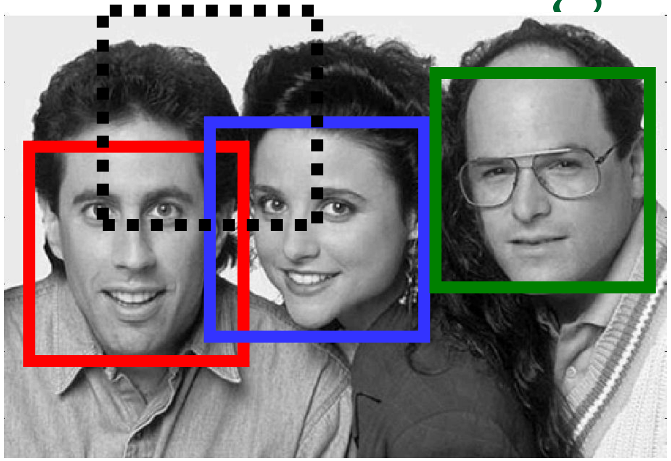
- What exactly is the "match"
  - What is the match "score"
- The DOT Product
  - Express the typical face as a vector
  - Express the region of the image being evaluated as a vector
    - But first histogram equalize the region
      - Just the section being evaluated, without considering the rest of the image
  - Compute the dot product of the typical face vector and the "region" vector

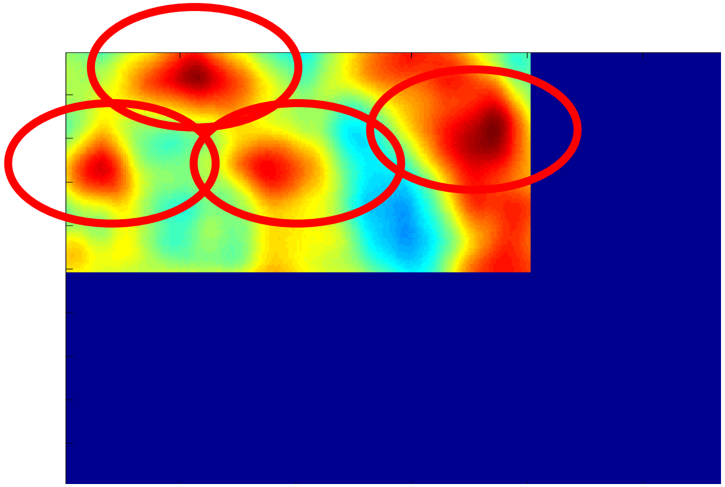
What do we get



- The right panel shows the dot product a various loctions
  - Redder is higher
    - The locations of peaks indicate locations of faces!

What do we get

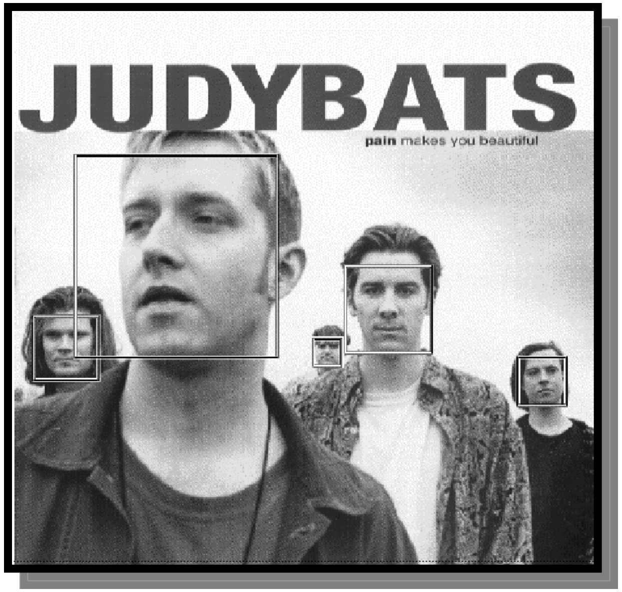




- The right panel shows the dot product a various loctions
  - Redder is higher
    - The locations of peaks indicate locations of faces!
- Correctly detects all three faces
  - Likes George’s face most
    - He looks most like the typical face
- Also finds a face where there is none!
  - A false alarm

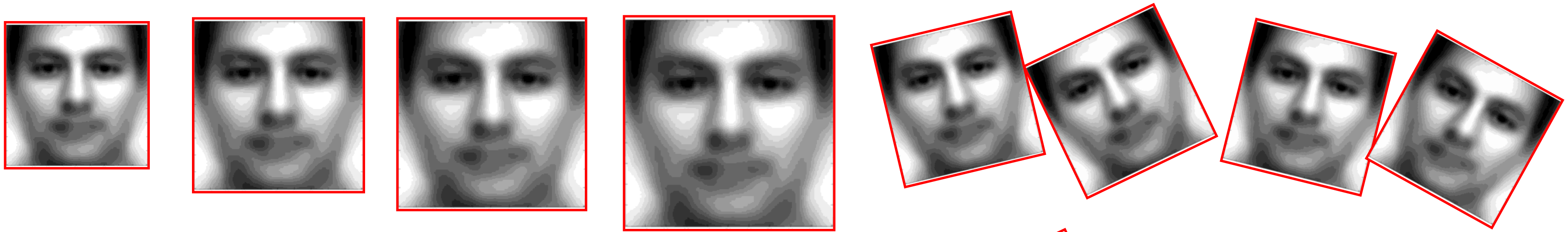
Scaling and Rotation Problems

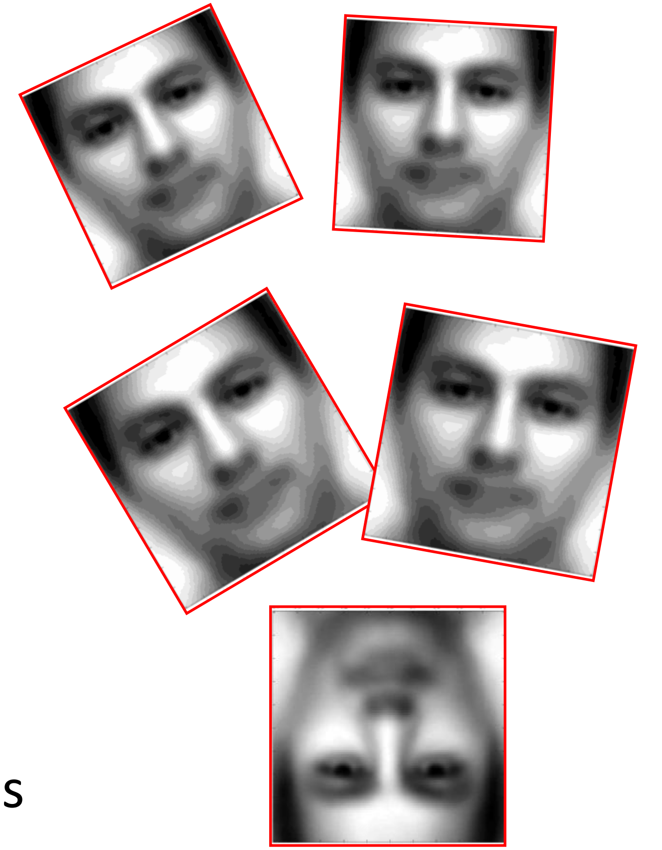
- Scaling
  - Not all faces are the same size
  - Some people have bigger faces
  - The size of the face on the image changes with perspective
  - Our "typical face" only represents one of these sizes



- Rotation
  - The head need not always be upright!
    - Our typical face image was upright

Solution





- Create many "typical faces"
  - One for each scaling factor
  - One for each rotation
    - How will we do this?
- Match them all

- Does this work
  - Kind of .. Not well enough at all
  - We need more sophisticated models

Face Detection: A Quick Historical Perspective

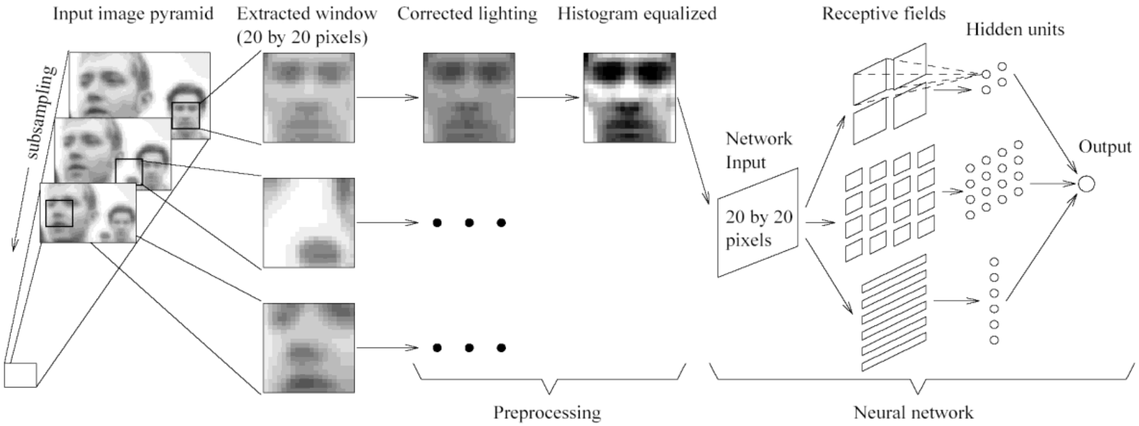


Figure 1: The basic algorithm used for face detection.

- Many more complex methods
  - Use edge detectors and search for face like patterns
  - Find "feature" detectors (noses, ears..) and employ them in complex neural networks..

- The Viola Jones method
  - Boosted cascaded classifiers

- Next in the program..