Machine Learning for Signal Processing

Deterministic Representations

Instructor: Bhiksha Raj

- Halo around CMU yesterday
  - Photo by Shikhar Agnihotri…

Index

- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

Index

- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

Representing Data

- The first and most important step in processing signals is representing them appropriately

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!"

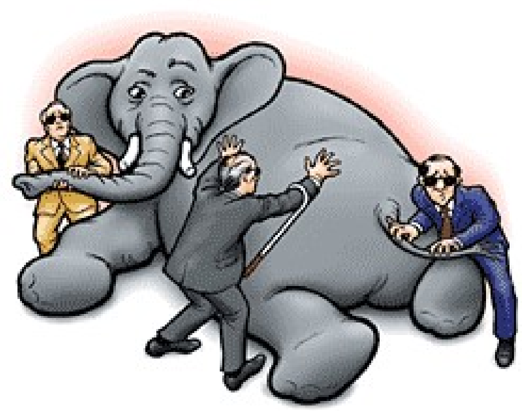
- The third approached the animal,
  And happening to take
  The squirming trunk within his hands,
  Thus boldly up and spake:
  "I see," quoth he, "the elephant
  Is very like a snake!"

- 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."

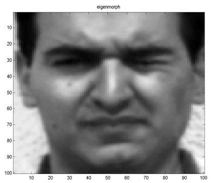
- The sixth no sooner had begun
  About the beast to grope,
  Then seizing on the swinging tail
  That fell within his scope,
  "I see," quoth he, "the elephant
  Is very like a rope."

- 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.

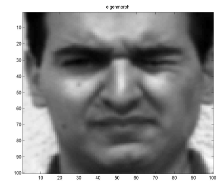


Representation

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




Representation




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


- More images



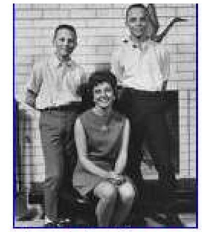
Still more images

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



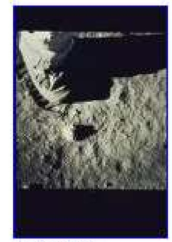
aboard **Apollo space** capsule.
1017 x 1280 - 130k
LIFE

Apollo Xi
1228 x 1280 - 181k
LIFE

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

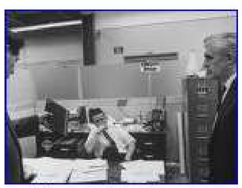
Splashdown of **Apollo** XI mission.
1280 x 866 - 184k
LIFE

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

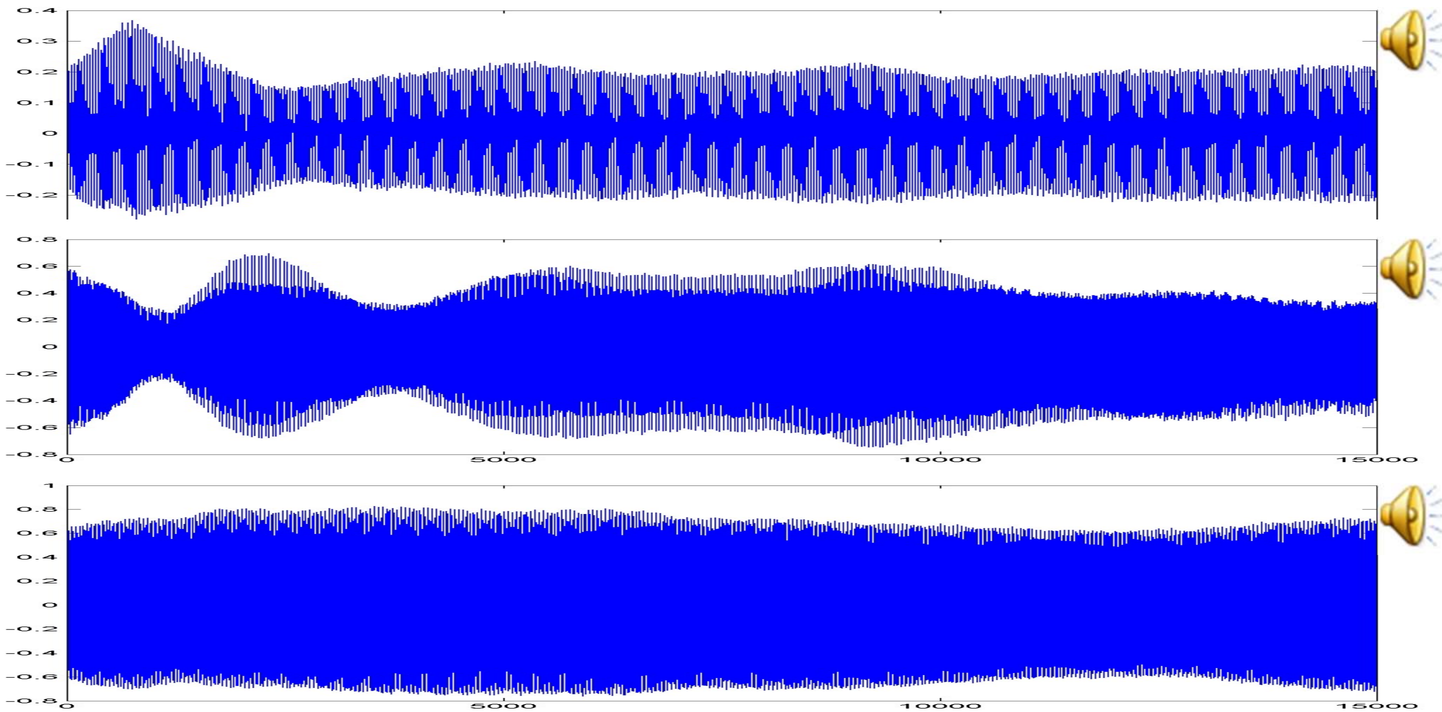
Apollo 8 Crew
968 x 1280 - 125k
LIFE

How do you describe them?

Representation

- Pixel-based descriptions are uninformative
  - Even though they enable perfect reconstruction
- Content-based descriptions are infeasible in the general case

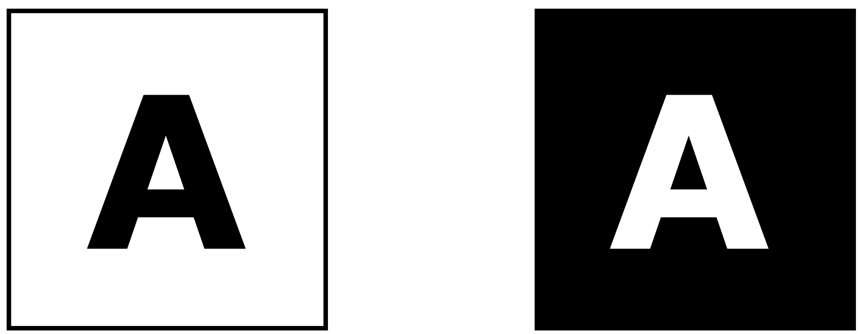
Sounds



- 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 won’t 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

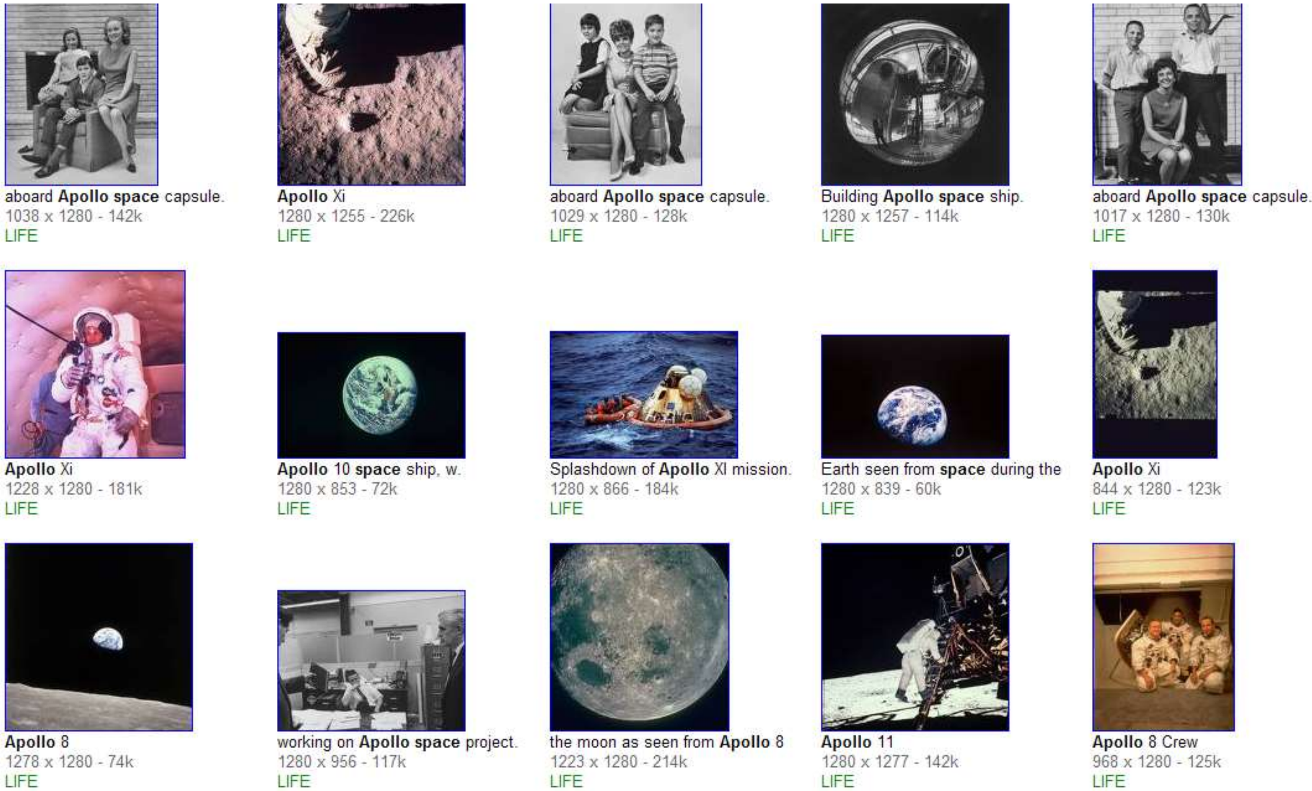


- Must be numerically representable
  – And reasonably universal
- Must allow identification, comparison, storage, reconstruction..

Index

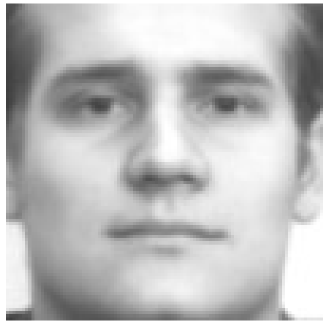
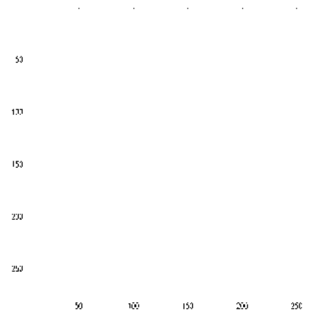
- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

Representing images



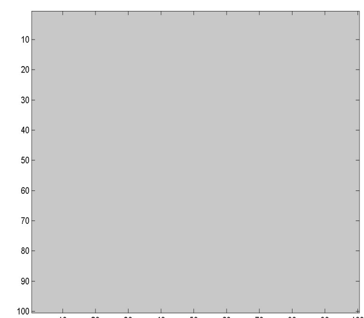
- The most common element in the image: background
  - Or rather large regions of relatively featureless shading
  - Uniform sequences of numbers

Representing images using a “plain” image

$$B = \begin{bmatrix} 1 \\ 1 \\ \cdot \\ 1 \end{bmatrix} \qquad \text{Image} = \begin{bmatrix} pixel\,1 \\ pixel\,2 \\ \cdot \\ pixel\,N \end{bmatrix}$$

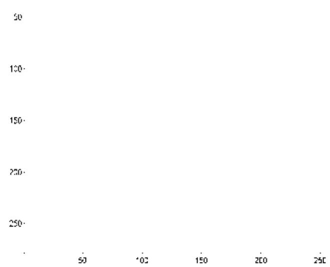
- Most of the figure is a more-or-less uniform shade
  - Dumb approximation – a image is a block of uniform shade
    - Will be mostly right!
- How to compute the “best” description? Projection
  - Represent the images as vectors and compute the projection of the image on the “basis”



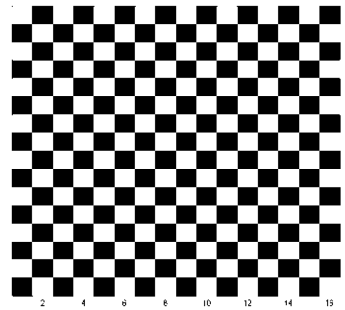
$$BW \approx Image$$
$$W = pinv(B)Image$$
$$PROJECTION = BW = Bpinv(B)Image$$

Adding more bases

B_1



B_2



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

- Let’s 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**

$$\mathrm{Im}age \approx w_1 B_1 + w_2 B_2$$

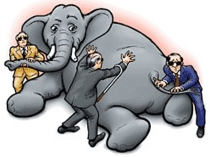
$$W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \qquad B = [B_1 \;\; B_2]$$

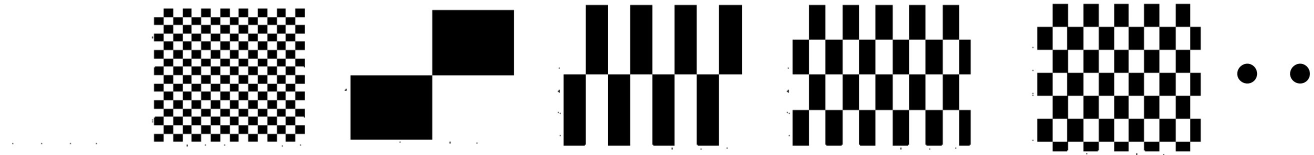
$$BW \approx Image$$

$$W = pinv(B)Image$$

$$PROJECTION = BW = Bpinv(B)Image$$

Adding still more bases





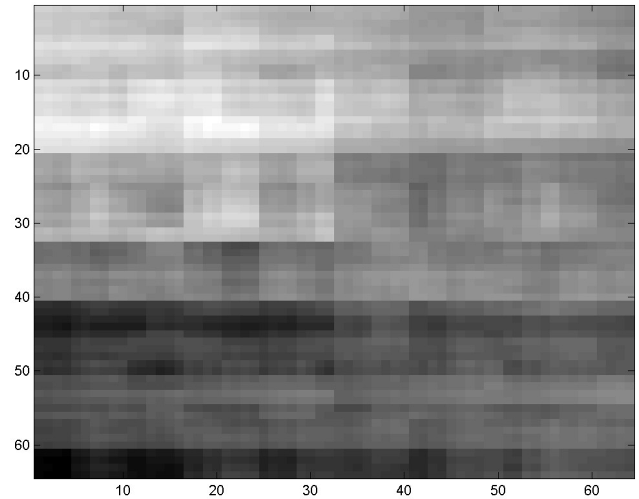
B1 B2 B3 B4 B5 B6 . .

- Regions that change with different speeds

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

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

$$BW \approx Image$$
$$W = pinv(B)Image$$
$$PROJECTION = BW = Bpinv(B)Image$$



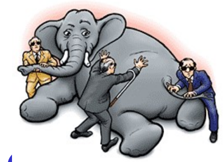
Getting closer at 625 bases!

Poll 1

- Mark the true statements.
  - The checkerboard patterns represent “bases” that are used to explain the images
  - Adding more varied checkerboard patterns increases our ability to represent images
  - We can add an unlimited number of checkerboards to continuously improve the representation

Poll 1

- Mark the true statements.
  - **The checkerboard patterns represent “bases” that are used to explain the images**
  - **Adding more varied checkerboard patterns increases our ability to represent images**
  - We can add an unlimited number of checkerboards to continuously improve the representation



Representation using checkerboards

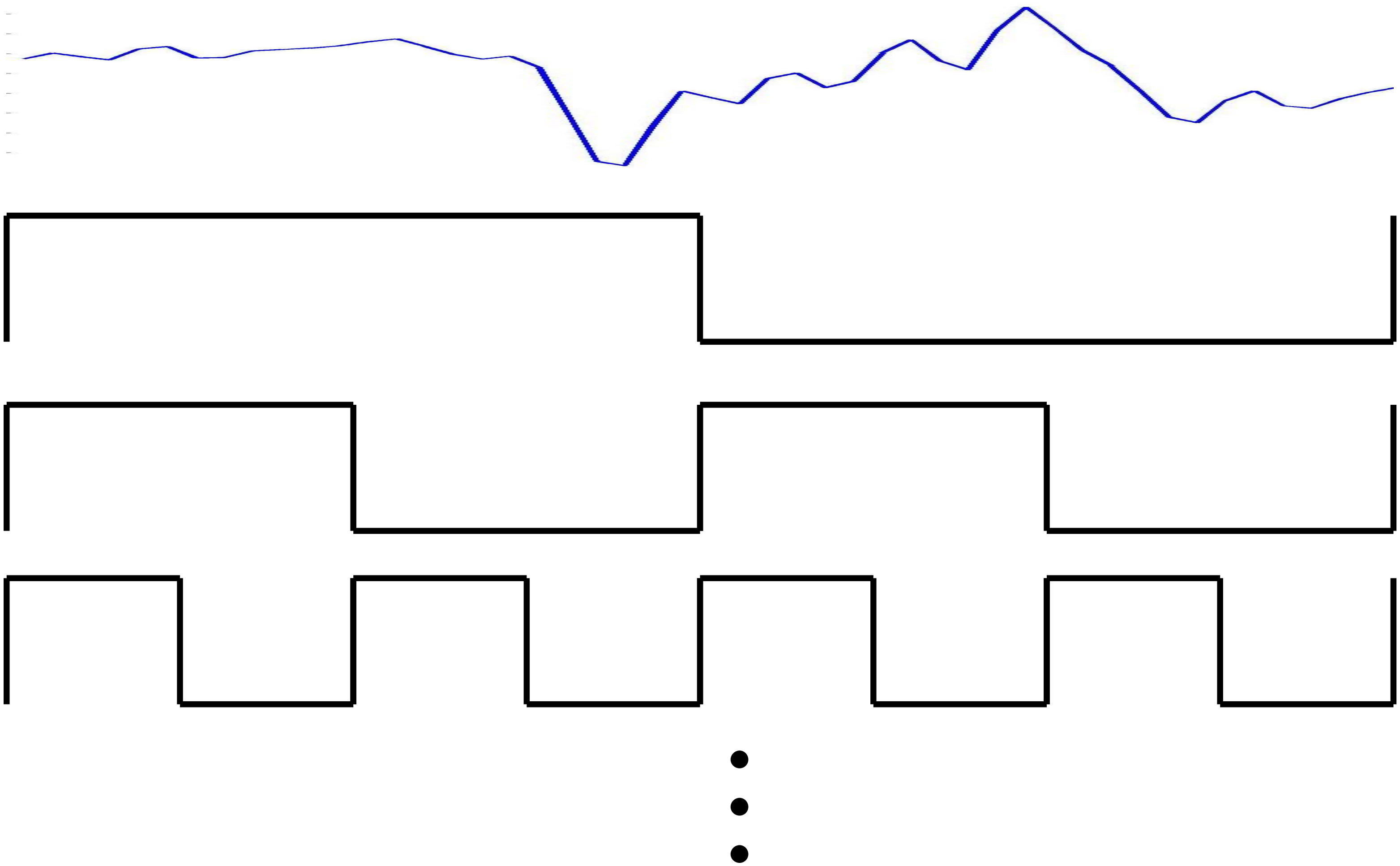
- A "standard" representation
  - Checker boards are the same regardless of the picture you're trying to describe
    - As opposed to using "nose shape" to describe faces and "leaf colour" to describe trees.

- Any image can be specified as (for example)

$$\text{Image} = 0.8\, checkerboard_0 + 0.2\, checkerboard_1 + 0.82 checkerboard_2 + \cdots$$

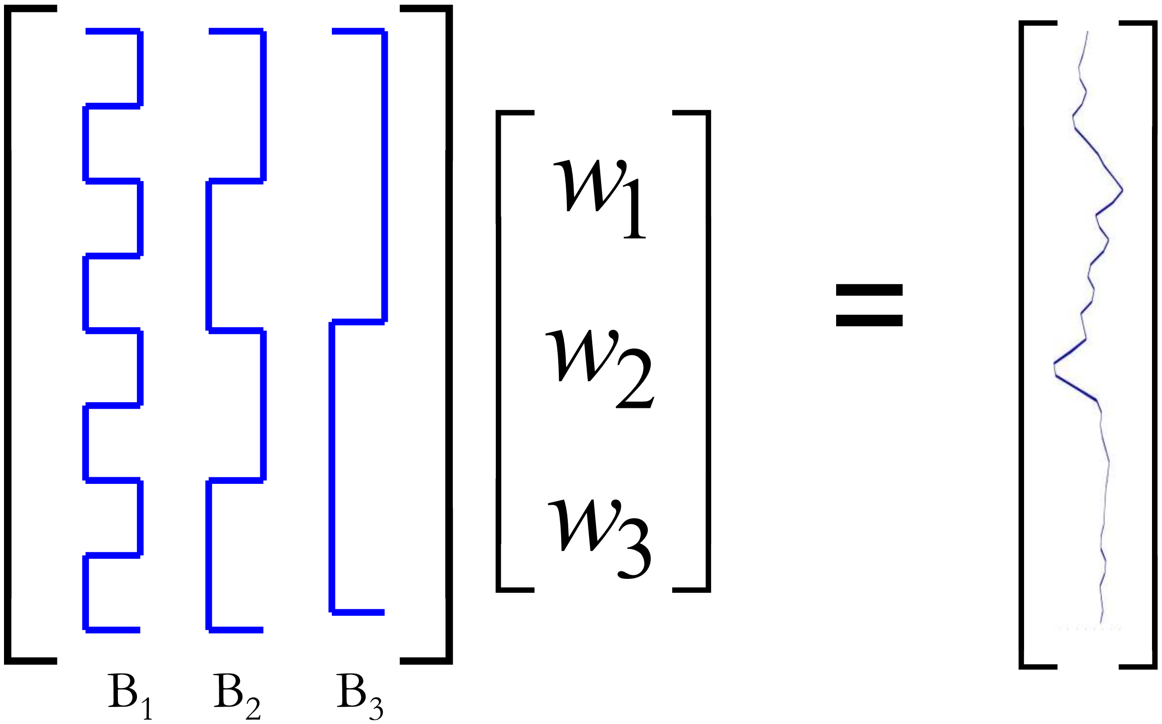
- The definition is sufficient to reconstruct the image to some degree
  - Not perfectly though

What about sounds?



- Square wave equivalents of checker boards

Projecting sounds



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

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

$$BW \approx Signal$$
$$W = pinv(B)Signal$$
$$PROJECTION = BW = Bpinv(B)Signal$$

General Philosophy of Representation

- Identify a set of *standard structures*
  - E.g. checkerboards
  - We will call these “bases”
- Express the data as a weighted combination of these bases
  - $X = w_1 B_1 + w_2 B_2 + w_3 B_3 + \ldots$
- Chose weights w_1, w_2, w_3.. for the best representation of X
  - I.e. the error between X and $\Sigma_i w_i B_i$ is minimized
  - The error is generally chosen to be $||X - \Sigma_i w_i B_i||^2$
- The weights w_1, w_2, w_3.. fully specify the data
  - Since the bases are known beforehand
  - Knowing the weights is sufficient to reconstruct the data

CRITERIA FOR "GOOD" BASES

Bases requirements

- Non-redundancy
  - Each basis must represent information *not* already represented by other bases
  - I.e. bases must be orthogonal
    - $<B_i, B_j> = 0$ for i != j
  - Mathematical benefit: can compute $w_i = <B_i,X>$
- Compactness
  - Must be able to represent most of X with fewest bases
  - Completeness: For D-dimensional data, need no more than D bases

Useful fact: Property of orthogonal bases

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \\ b_{13} & b_{23} & b_{33} \end{bmatrix}$$

- The inverse of a matrix whose columns (rows) are unit length and orthogonal to one another is its transpose
- If the columns (rows) are not unit length (but still orthogonal), the inverse is still a transpose, but with the rows (columns) scaled by the squared length of the column vectors
- This is also true for non-square matrices: The pseudo inverse is just the transpose
  - With scaled rows, if the original columns are not unit length

Bases based representation

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

- Place all bases in basis matrix B

$$BW \approx X$$
$$W = Pinv(B)X$$

- For orthogonal bases

$$w_i = \frac{< B_i, X >}{\| B_i \|^2}$$

Poll 2

- The pseudo-inverse of an orthonormal matrix consisting of orthogonal unit vectors is its own transpose, true or false
  – T
  – F

- Mark true statements
  – If all bases are orthogonal to one another, the weights of the bases to compose a given vector can be determined individually for each basis
  – The weights with which the bases combine to compose a vector can be computed individually for each basis even if the bases are not orthogonal

Poll 2

- The pseudo-inverse of an orthonormal matrix consisting of orthogonal unit vectors is its own transpose, true or false
  - **T**
  - F

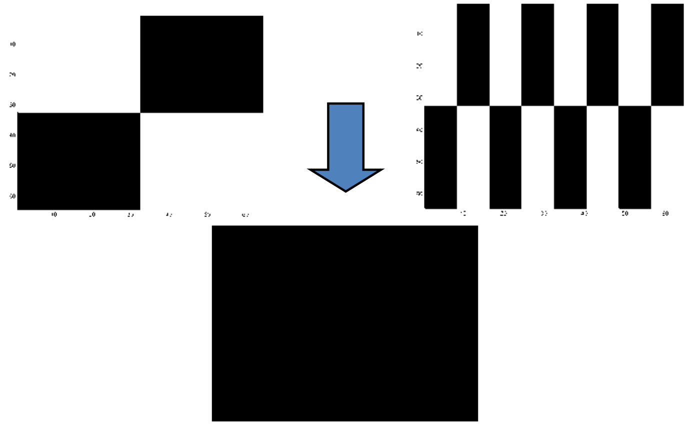
- Mark true statements
  - **If all bases are orthogonal to one another, the weights of the bases to compose a given vector can be determined individually for each basis**
  - The weights with which the bases combine to compose a vector can be computed individually for each basis even if the bases are not orthogonal

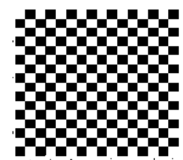
Bases based representation

- Challenge: Choice of appropriate bases

Why checkerboards are great bases

- We cannot explain one checkerboard in terms of another
  - The two are orthogonal to one another!
- This means we can determine the contributions of individual bases separately
  - Joint decomposition with multiple bases gives the same result as separate decomposition with each
  - This never holds true if one basis can explain another





$$B = \begin{array}{c} \\ \\ \end{array}\begin{array}{cc} B_1 & B_2 \\ \end{array}$$

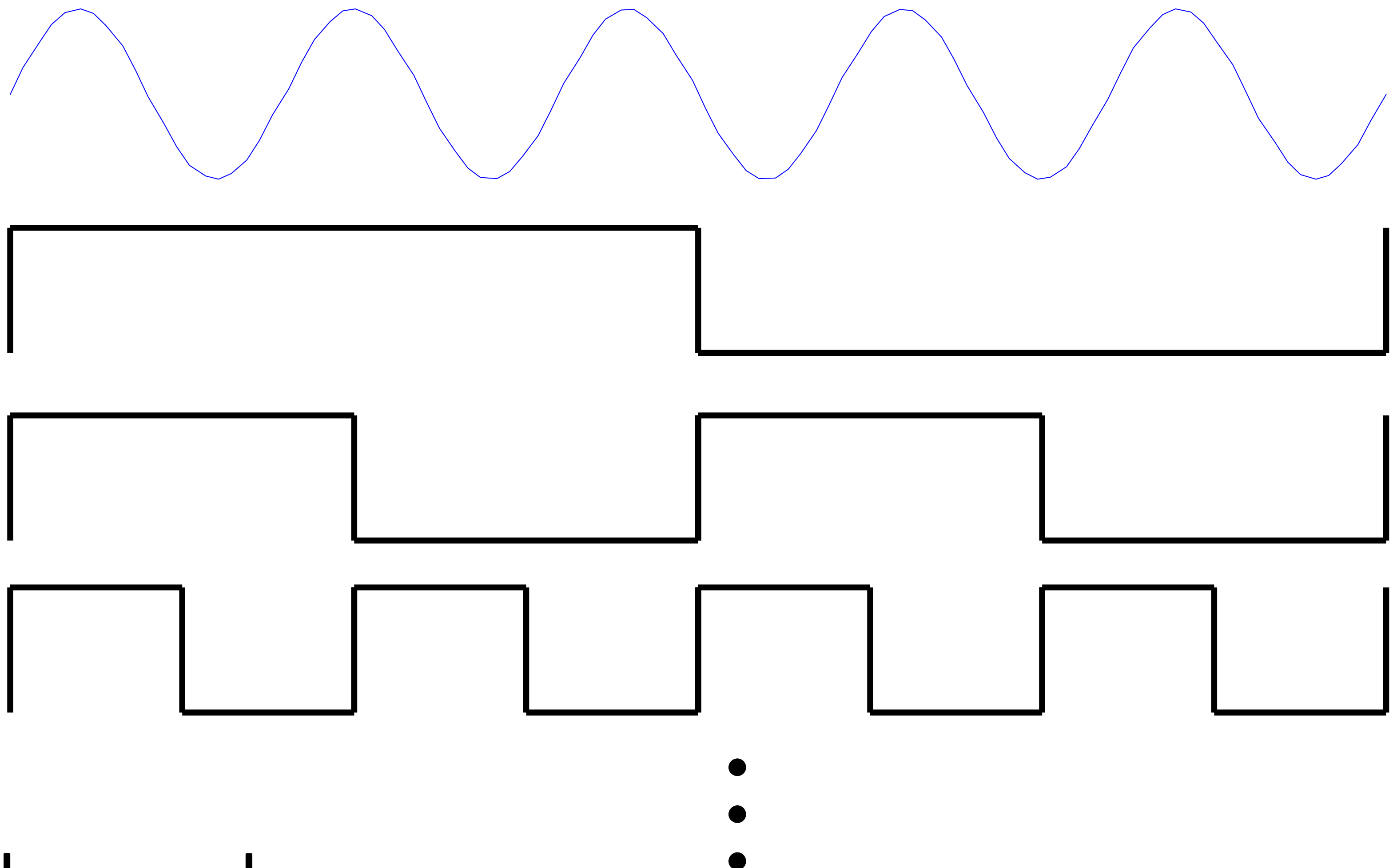
$$B = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix}$$

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

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

$$w_i = \frac{\langle B_i, Image \rangle}{\|B_i\|^2}$$

Checker boards are not good bases

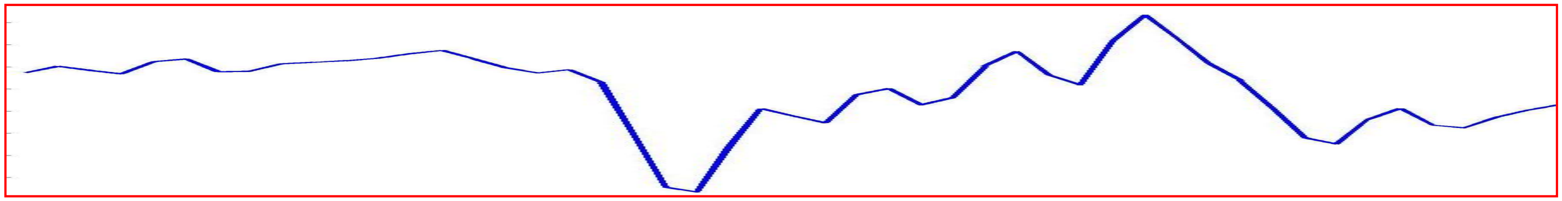


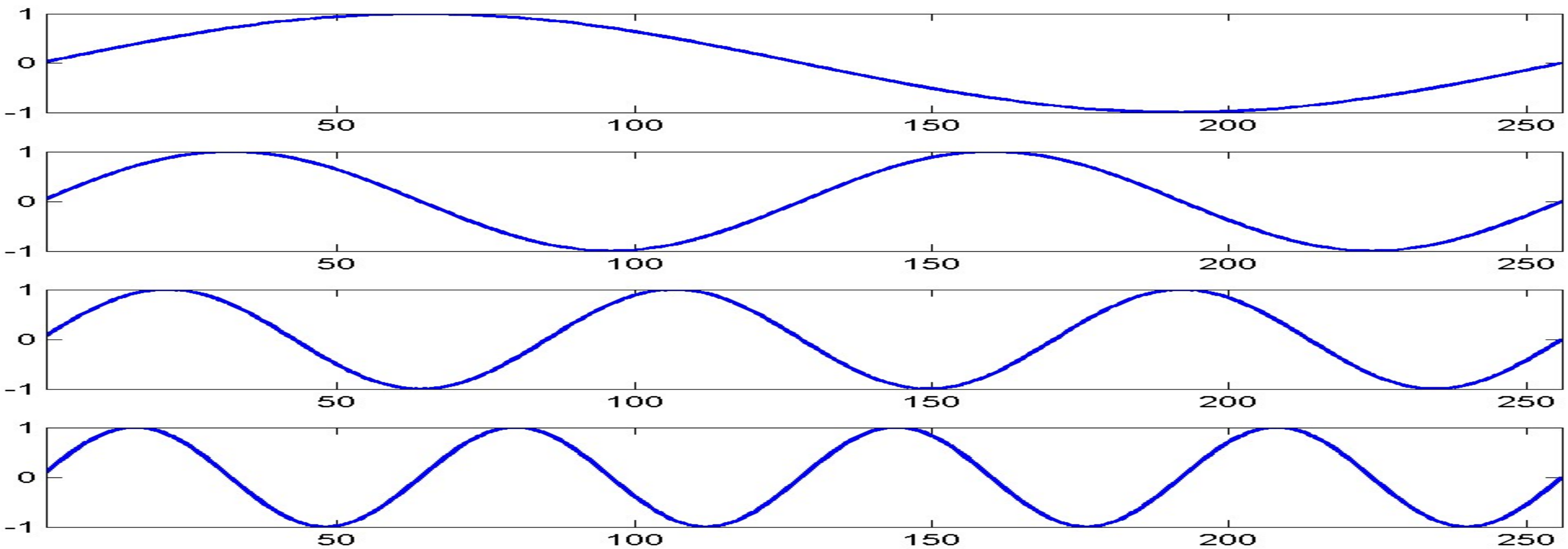
- Sharp edges
  - Can *never* be used to explain rounded curves

Index

- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

Sinusoids ARE good bases

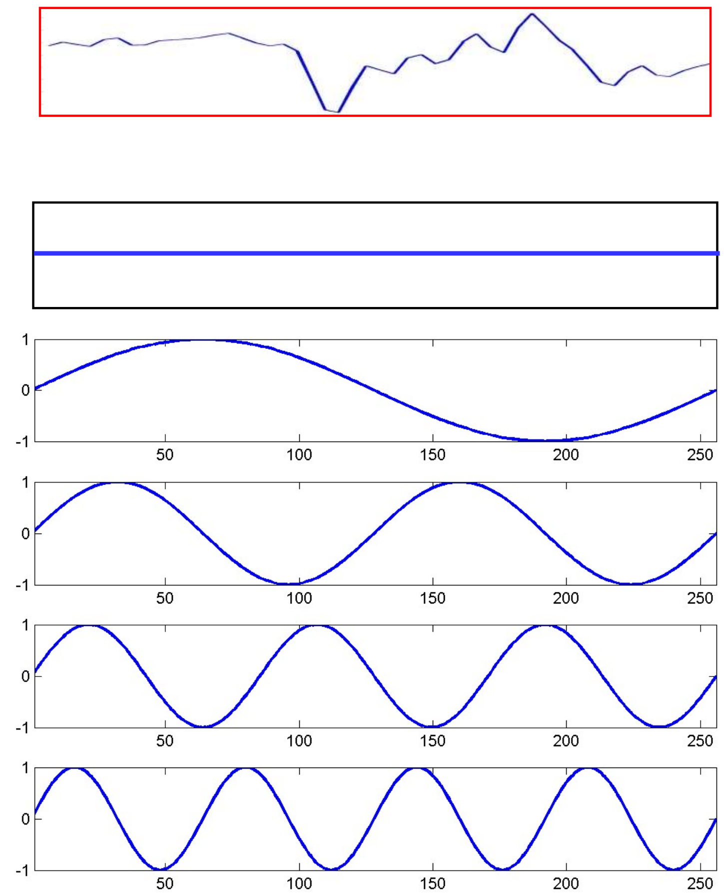




- They are orthogonal
- They can represent rounded shapes nicely
  - Unfortunately, they cannot represent sharp corners

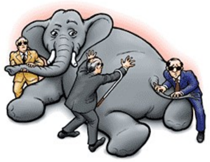
What are the frequencies of the sinusoids

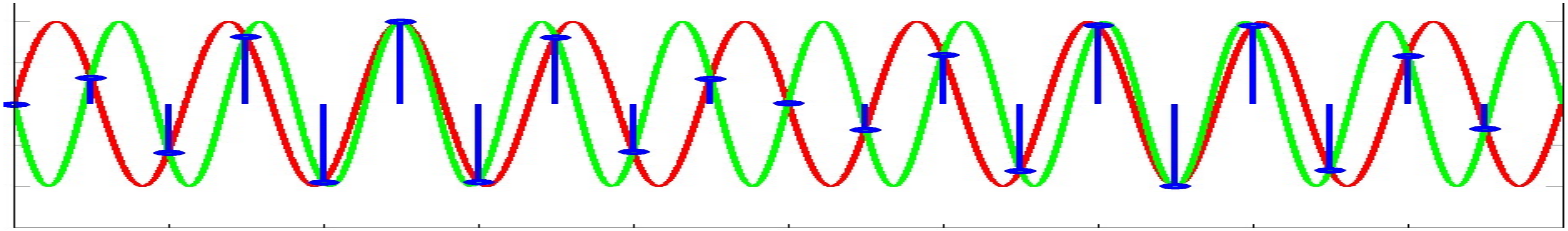
- Follow the same format as the checkerboard:
  - DC
  - The entire length of the signal is one period
  - The entire length of the signal is two periods.
- And so on..



- The k-th sinusoid:
  - $F(n) = \sin(2\pi kn/N)$
    - N is the length of the signal
    - k is the number of periods in N samples

How many frequencies in all?

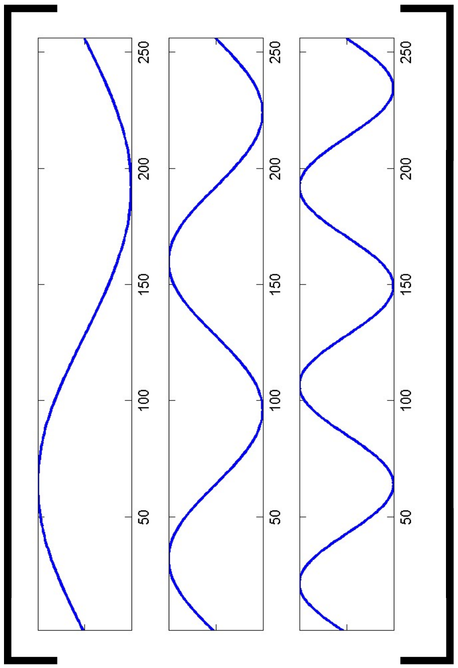




- A max of L/2 periods are possible
- If we try to go to (L/2 + X) periods, it ends up being identical to having (L/2 – X) periods
  - With sign inversion

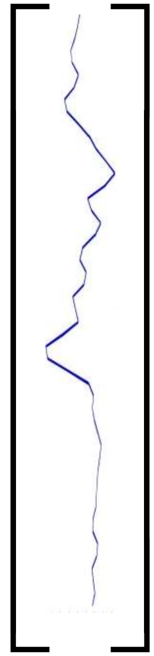
- Example for L = 20
  - Red curve = sine with 9 cycles (in a 20 point sequence)
    - $Y(n) = \sin(2\pi 9n/20)$
  - Green curve = sine with 11 cycles in 20 points
    - $Y(n) = -\sin(2\pi 11n/20)$
  - The blue lines show the actual samples obtained
    - These are the only numbers stored on the computer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

How to compose the signal from sinusoids



$B_1 \quad B_2 \quad B_3$

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} =$$



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

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

$$BW \approx Signal$$
$$W = Pinv(B)Signal$$
$$Projection = (B(B^T B)^{-1} B^T)\, Signal$$

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

How to compose the signal from sinusoids

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

L/2 columns only

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

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

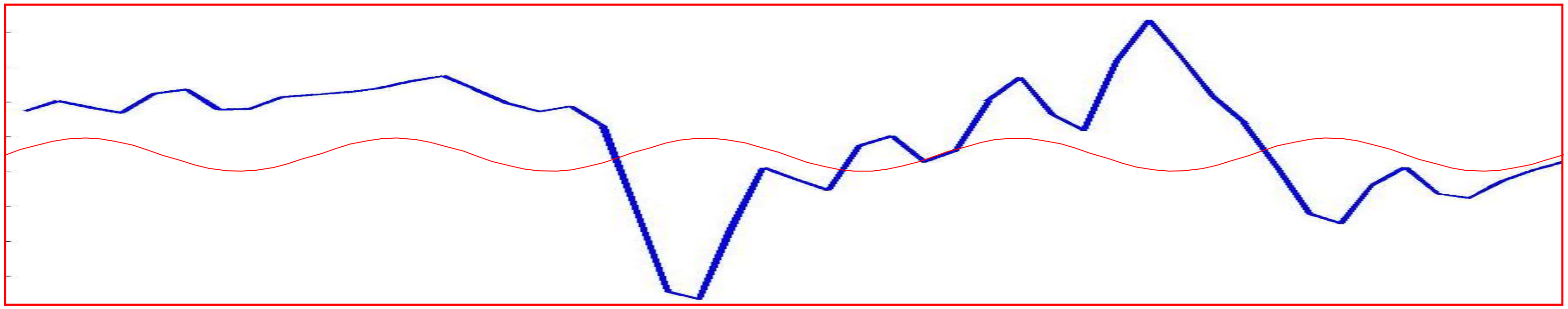
$$Signal = \begin{bmatrix} s[0] \\ s[1] \\ . \\ s[L-1] \end{bmatrix}$$

$$BW \approx Signal$$

$$W = pinv(B)Signal$$

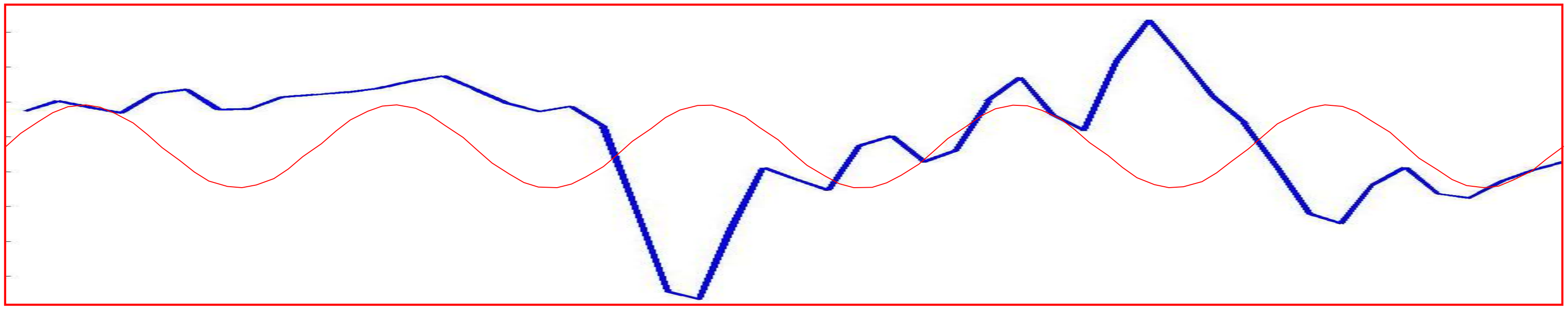
- The sines form the vectors of the projection matrix
  - Pinv() will do the trick as usual

Interpretation..



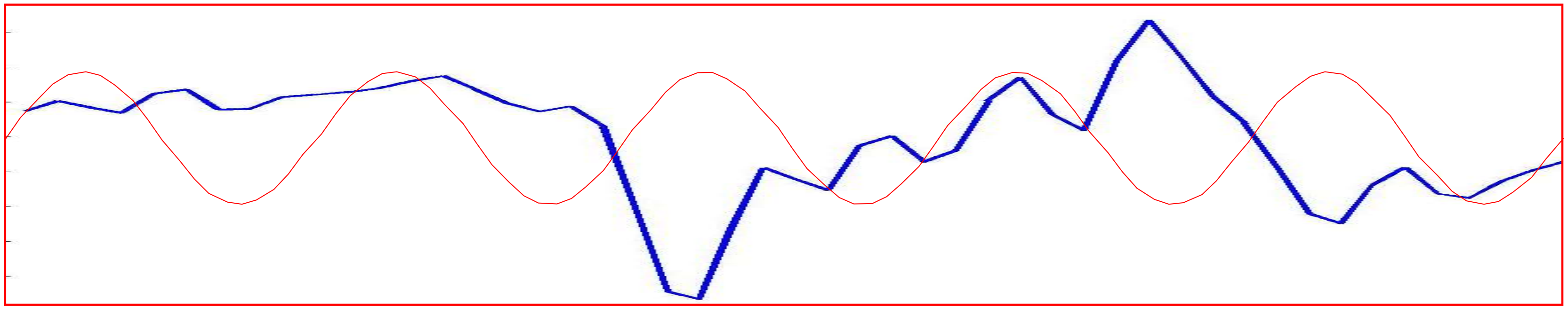
- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..



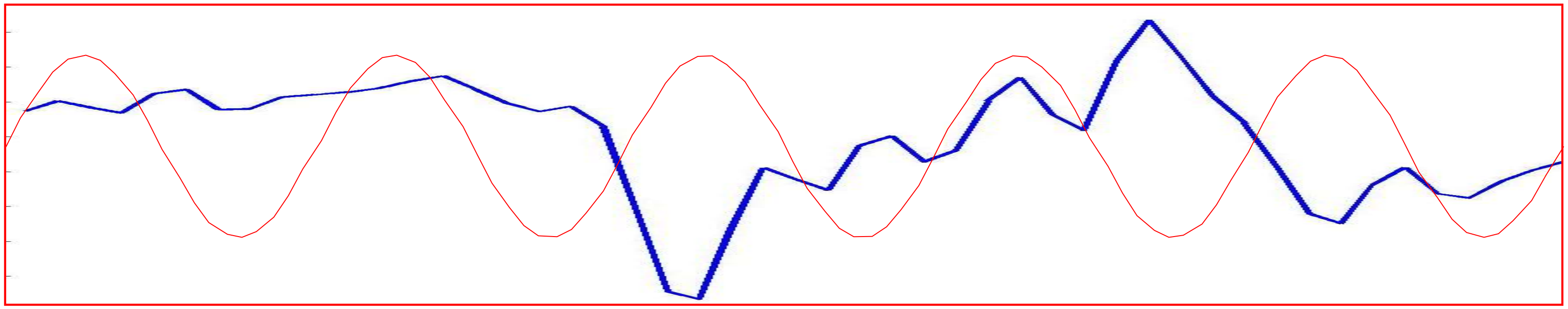
- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..



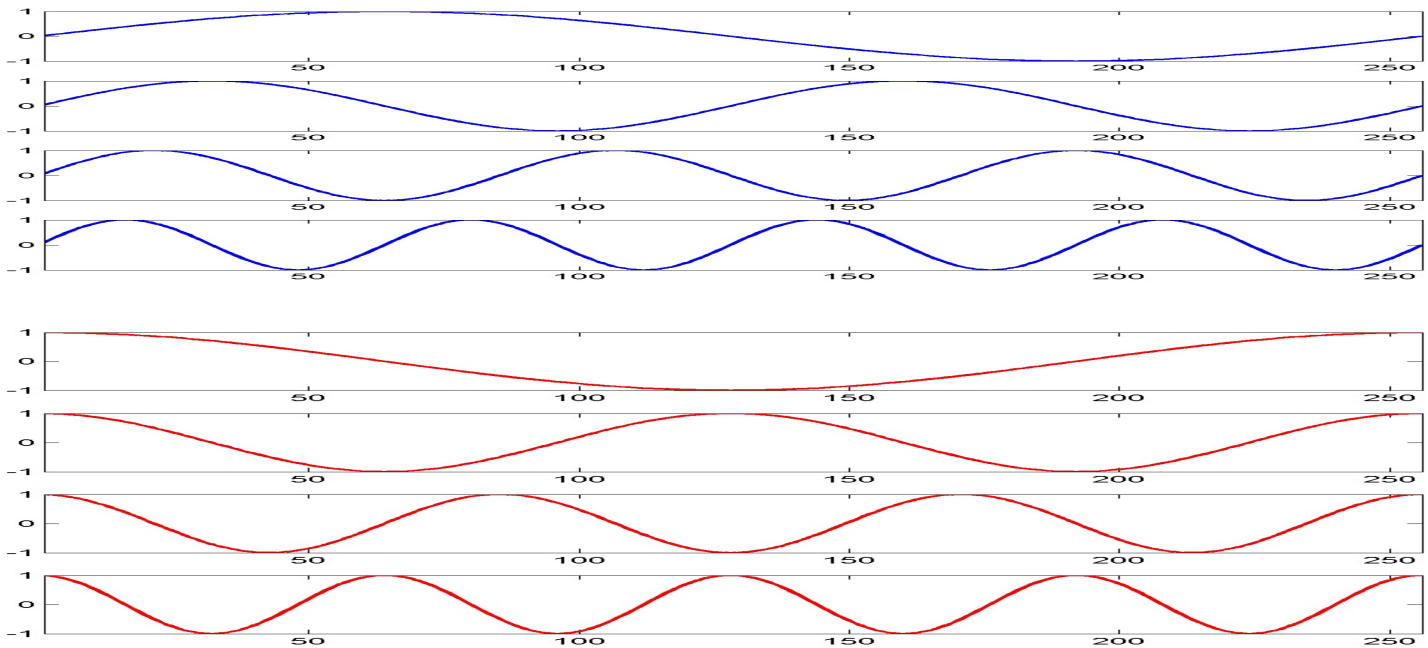
- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..

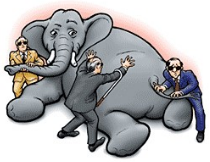


- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

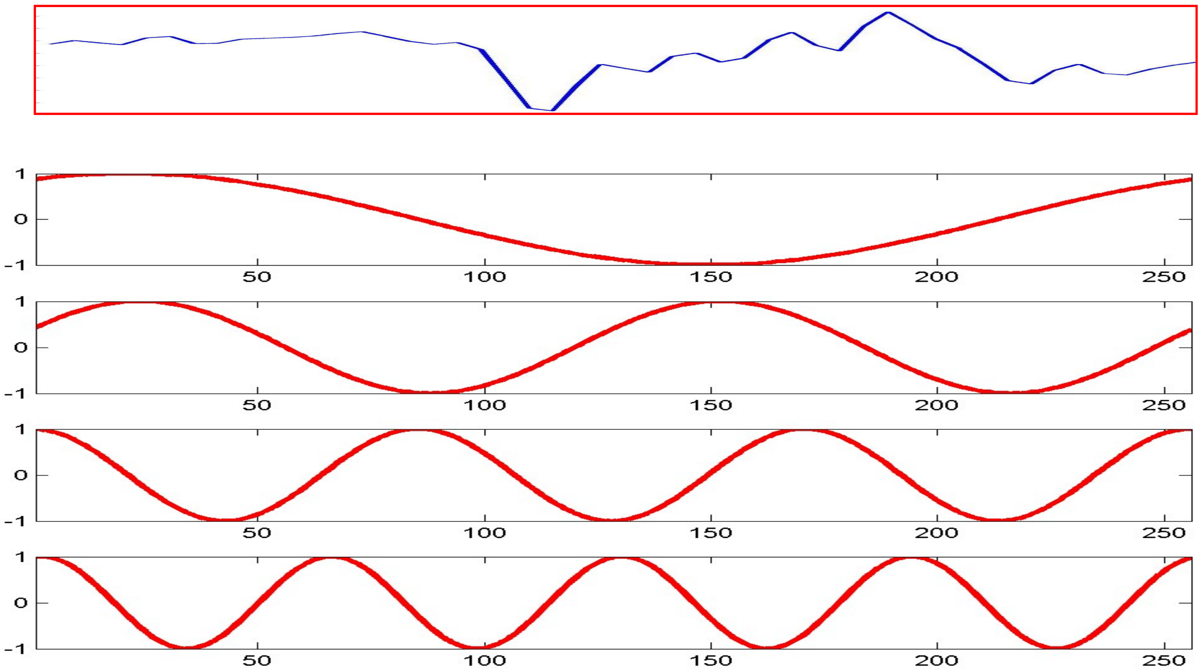
Sines by themselves are not enough



- 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



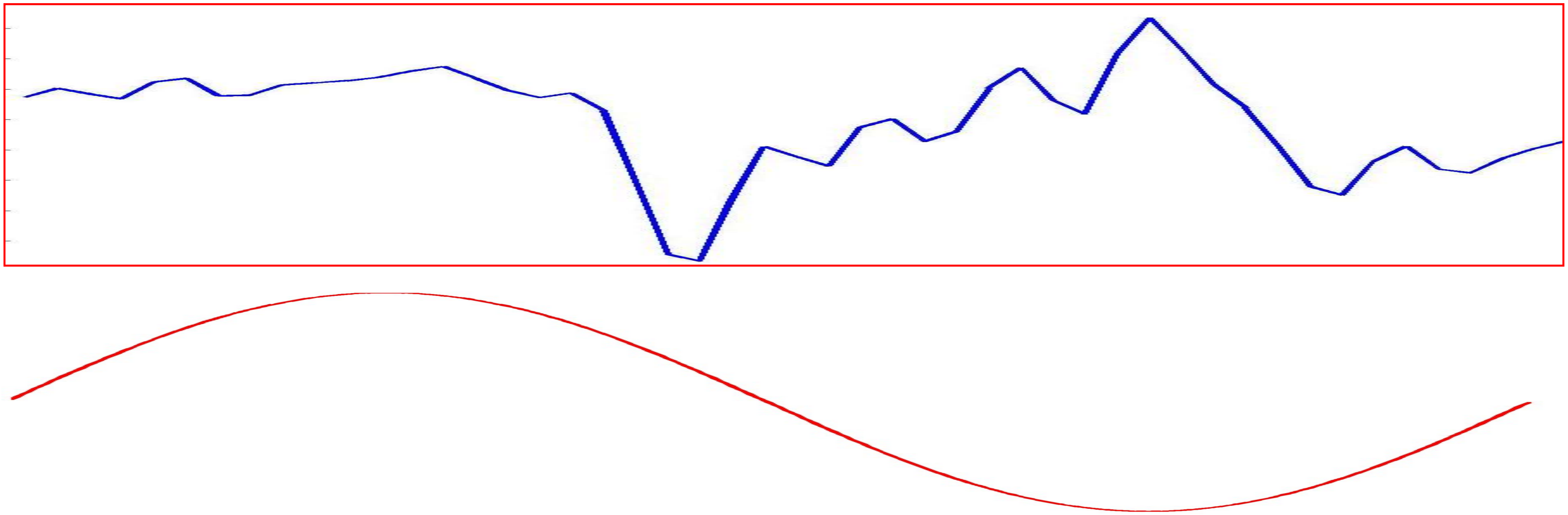
Sines are shifted: do not start with value = 0

- Allow the sinusoids to move!

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

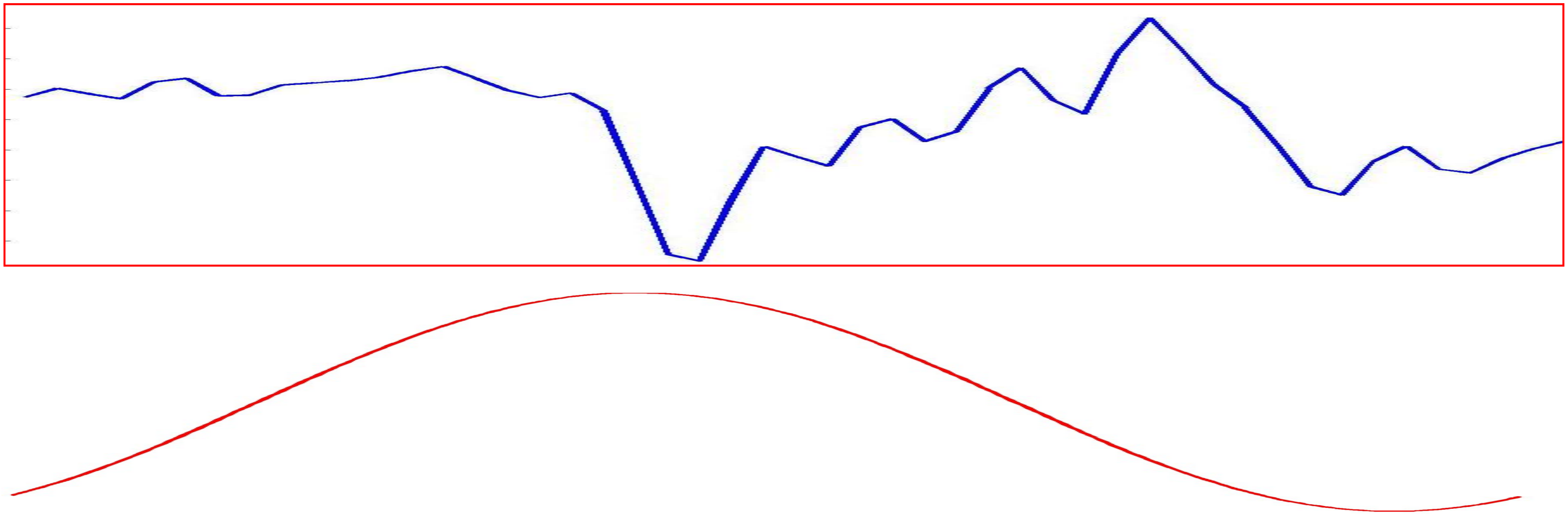
- How much do the sines shift?

Determining phase



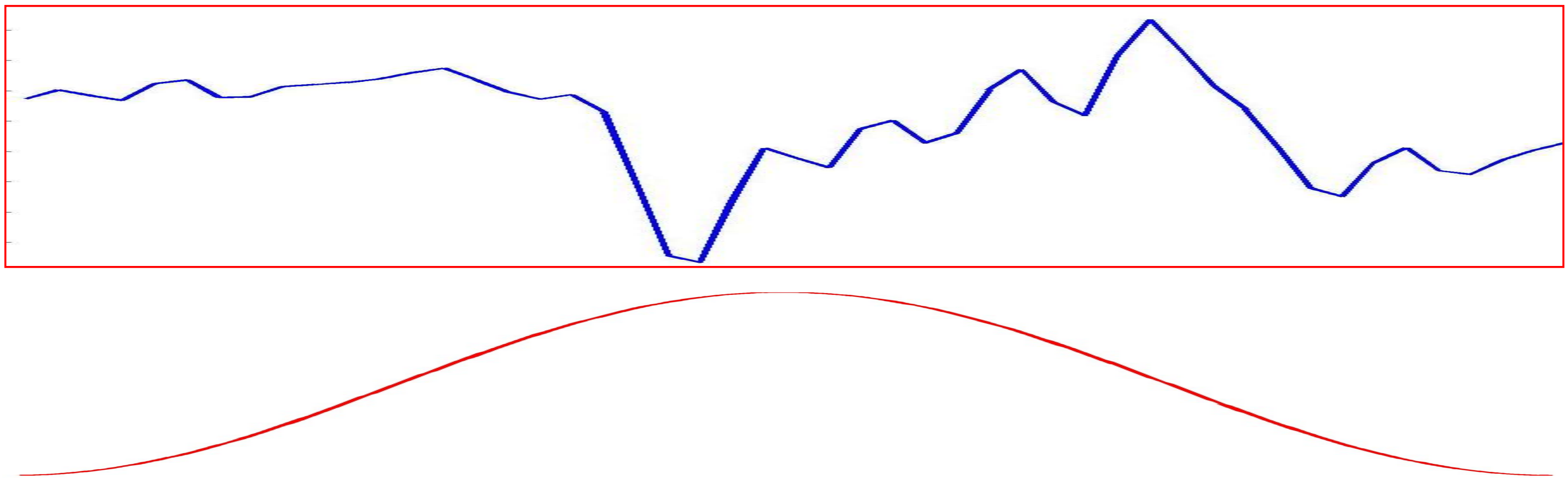
- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

Determining phase



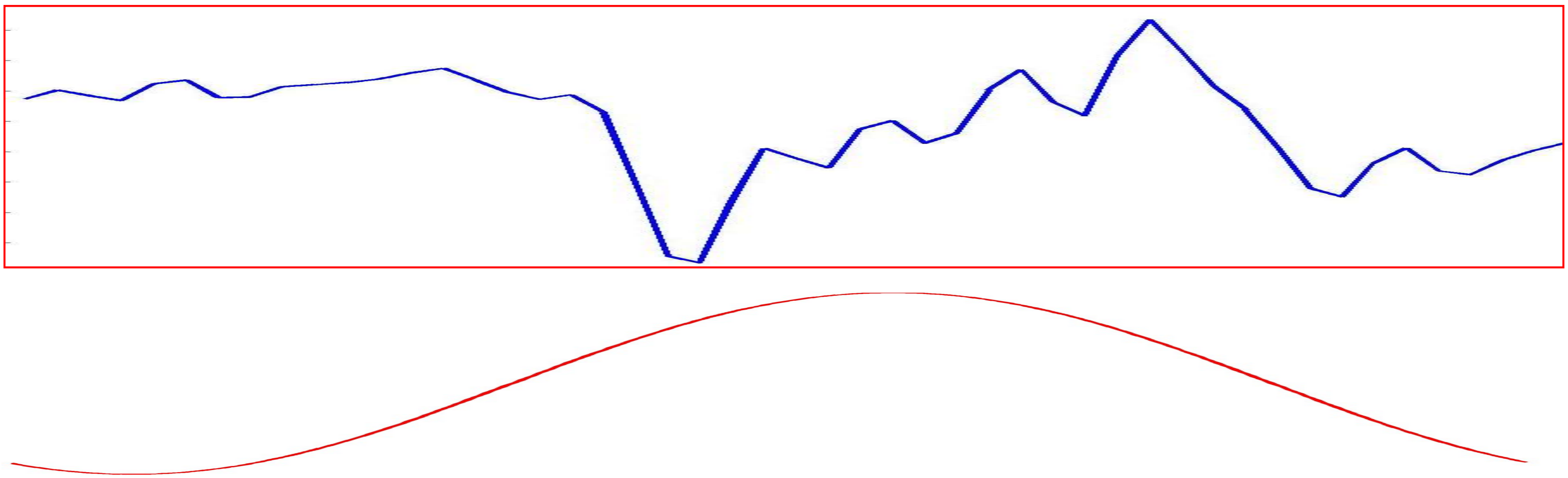
- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

Determining phase

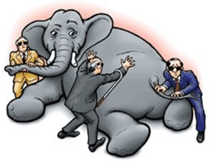


- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

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



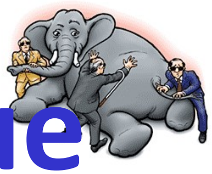
The problem with phase

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

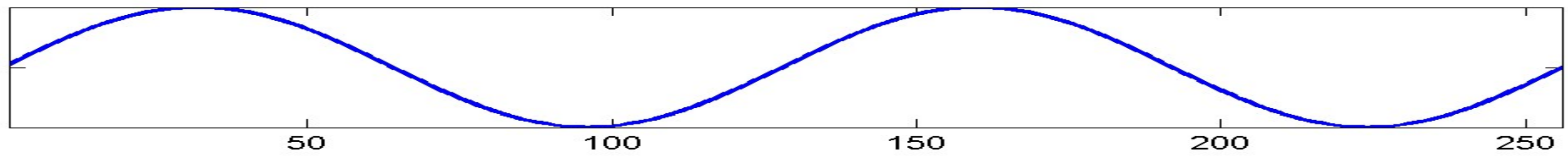
L/2 columns only

- This can no longer be expressed as a simple linear algebraic equation
  - The "basis matrix" depends on the unknown phase
    - 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

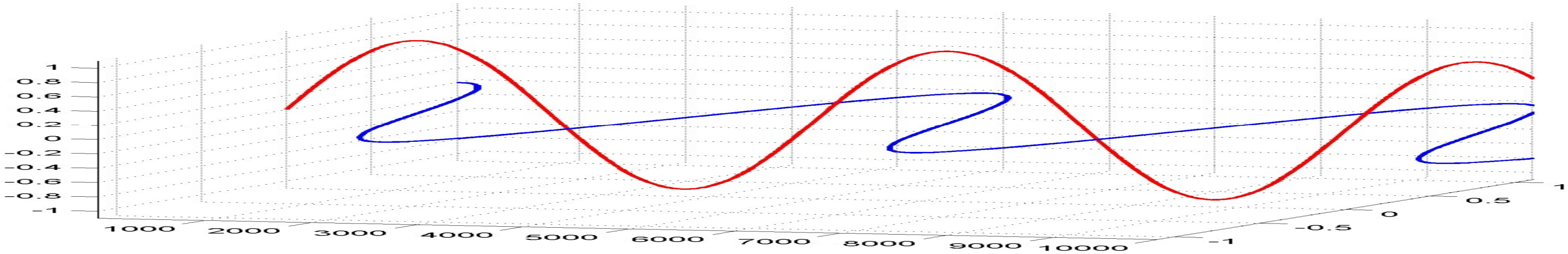


$$b[n] = sin(freq \times n)$$



$$b[n] = exp(j.freq.n) = \cos(freq.n) + jsin(freq.n)$$

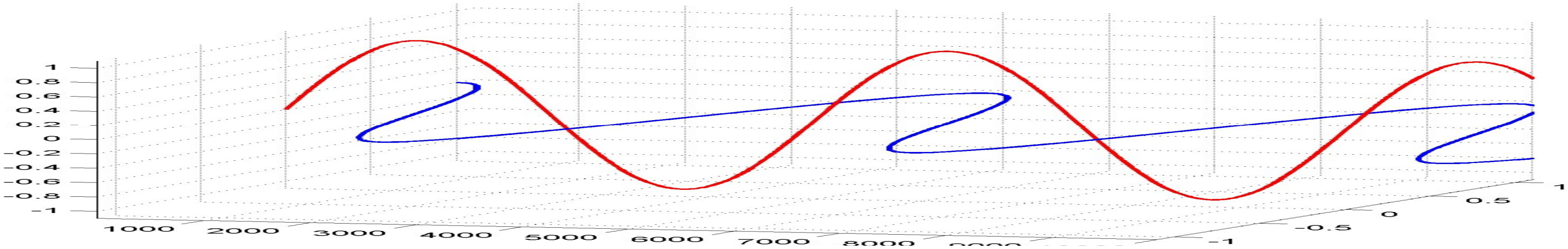
$$j = \sqrt{-1}$$



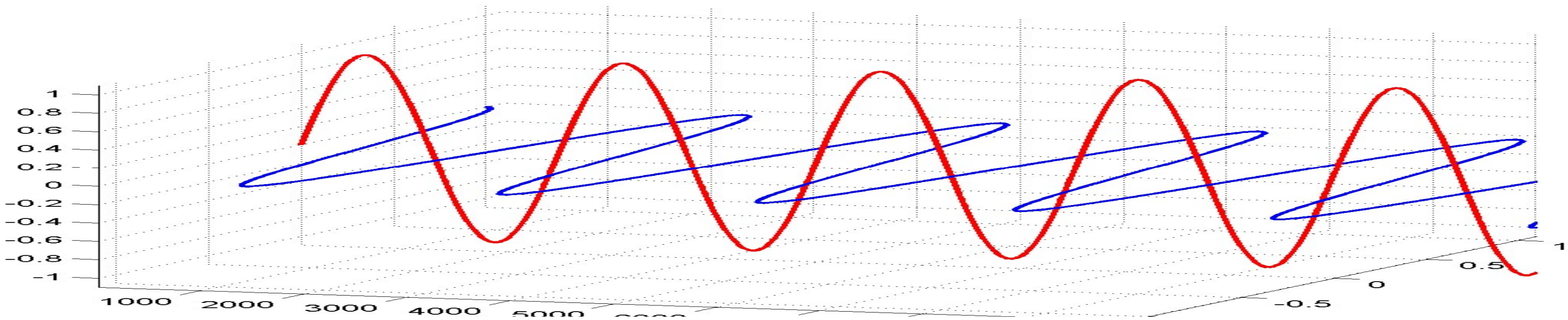
$$exp(j.(freq.n + \varphi)) = exp(j.freq.n)\exp(j\varphi)$$
$$= \cos(freq.n + \varphi) + jsin(freq.n + \varphi)$$

- 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!!

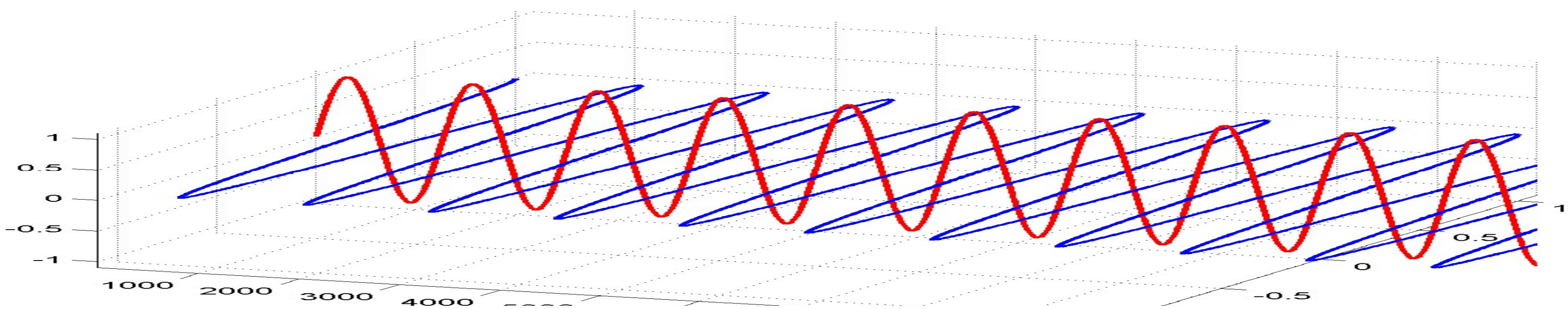
Explaining with Complex Exponentials

A x 

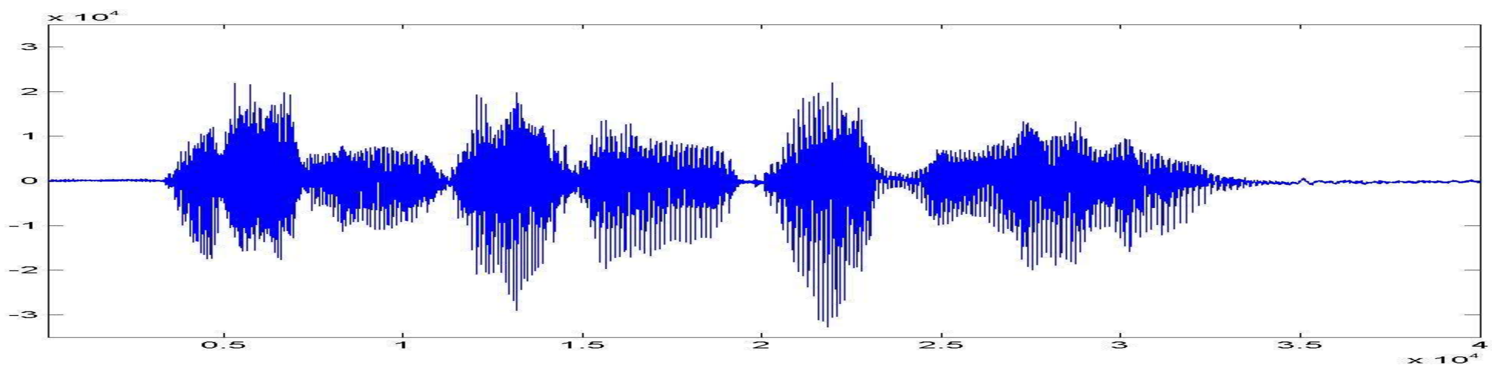
+

B x 

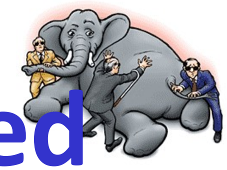
+

C x 

=



Complex exponentials are well behaved



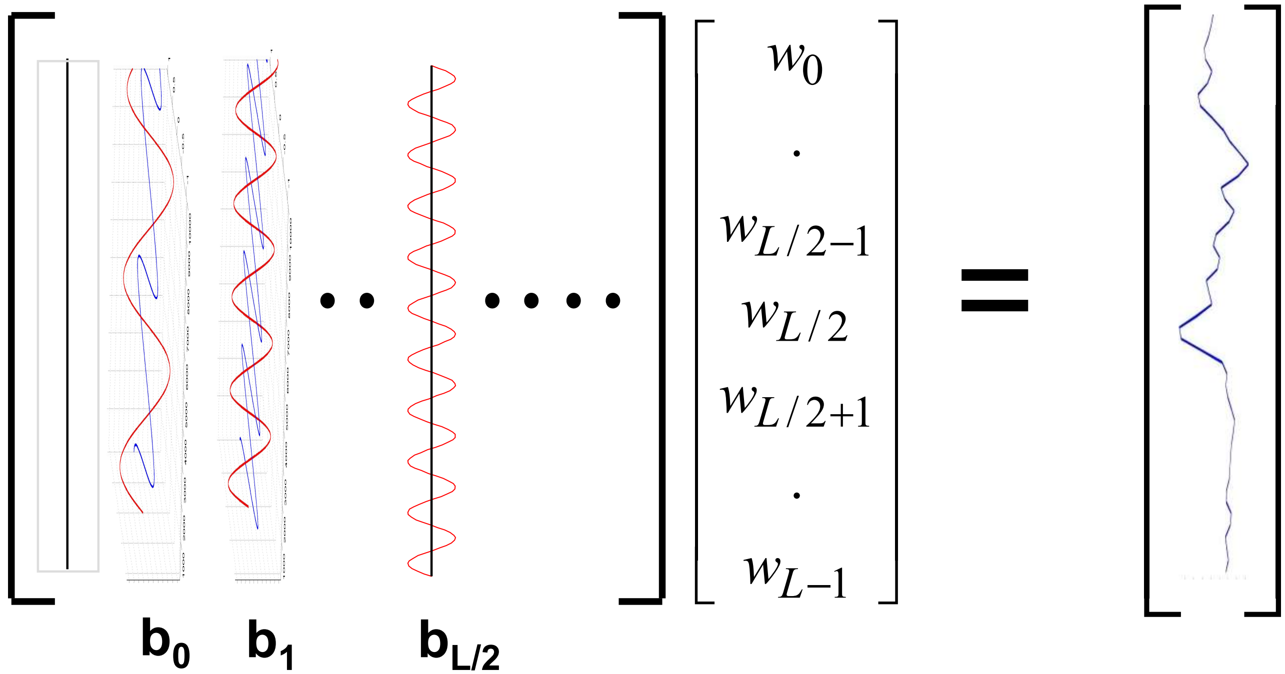
- Like sinusoids, a complex exponential of one frequency can never explain one of another frequency
  - They are orthogonal

- 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
    - $\cos(x) + j\sin(x) + \cos(x) - j\sin(x) = 2\cos(x)$

Complex Exponential bases



- Explain the data using L complex exponential bases

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: The basis do not include phase
  - The phase is obtained through a multiplicative term $\exp(j\varphi)$ which factors into S and is estimated

Poll 3

- Mark the true statements
  - Complex exponentials enable us to use linear algebraic math to express signals as linear combinations of phase-shifted sinusoids
  - Complex exponentials can be combined to construct purely real signals
  - Combinations of complex exponentials will always be complex

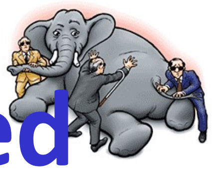
- How many unique complex exponentials (of different frequencies) do we need to construct an L-length discrete signal?
  - L
  - L/2
  - 2L

Poll 3

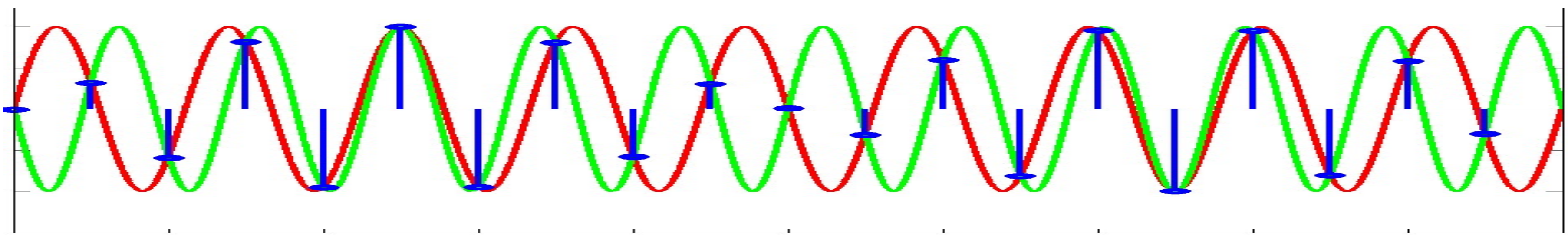
- Mark the true statements
  - **Complex exponentials enable us to use linear algebraic math to express signals as linear combinations of phase-shifted sinusoids**
  - **Complex exponentials can be combined to construct purely real signals**
  - Combinations of complex exponentials will always be complex

- How many unique complex exponentials (of different frequencies) do we need to construct an L-length discrete signal?
  - **L**
  - L/2
  - 2L

Complex exponentials are well behaved



- Conjugate symmetry
  - $\exp\left(j2\pi\frac{(L/2-x)n}{L}\right)+\exp\left(j2\pi\frac{(L/2+x)n}{L}\right)$ is real
    - The complex exponentials with frequencies equally spaced from L/2 are complex conjugates



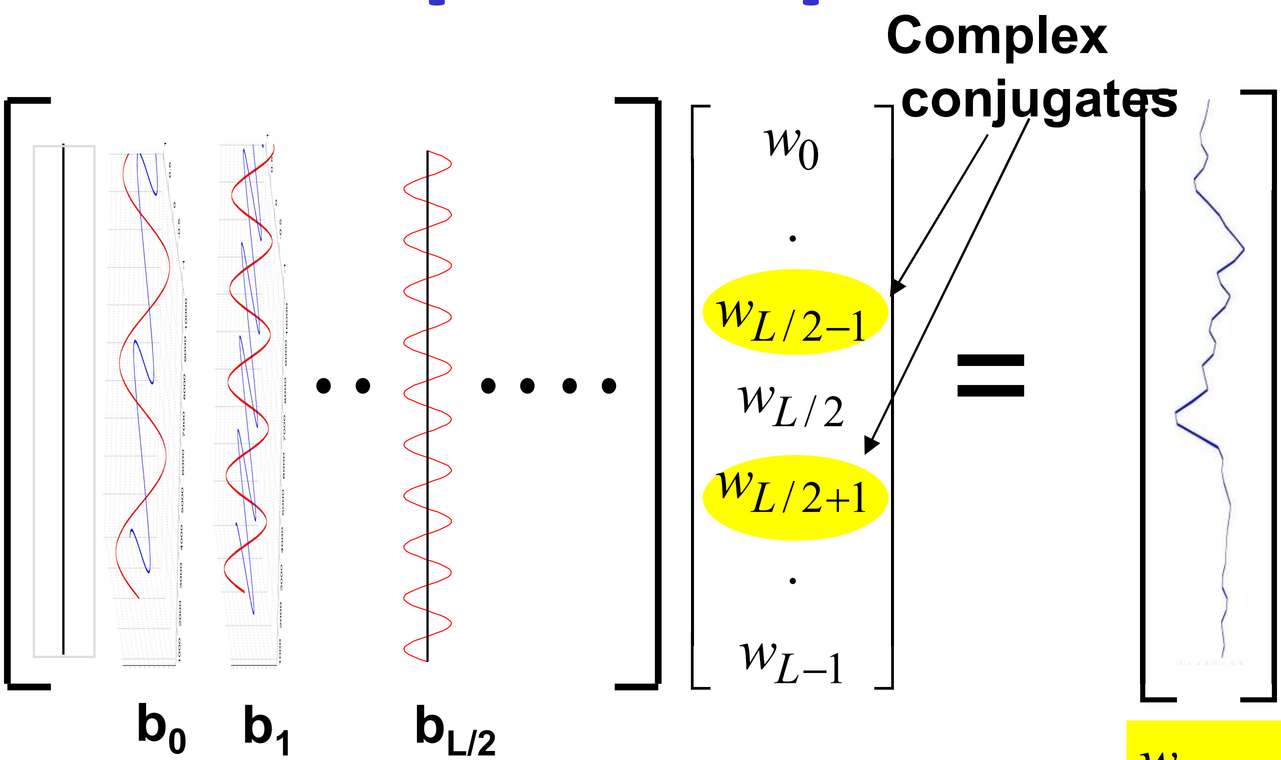
Complex exponentials are well behaved

- $$\exp\left(j2\pi\frac{(L/2-x)n}{L}\right)+\exp\left(j2\pi\frac{(L/2+x)n}{L}\right)$$
  - The complex exponentials with frequencies equally spaced from L/2 are complex conjugates
    - “Frequency = k” ➔ k periods in L samples

$$a\exp\left(j2\pi\frac{(L/2-x)n}{L}\right)+conjugate(a)\exp\left(j2\pi\frac{(L/2+x)n}{L}\right)$$

  - Is also real
  - If the two exponentials are multiplied by numbers that are conjugates of one another the result is real

Complex Exponential bases



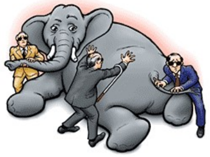
$$w_{L/2+k} = conjugate(w_{L/2-k})$$

- **For real signals:**
- The weights given to the (L/2 + k)th basis and the (L/2 – k)th basis should be complex conjugates, to make the result real
- Fortunately, a least squares fit will give us complex conjugate weights to both bases automatically

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: The basis
- Note that $S_{L/2+x} = \text{conjugate}(S_{L/2-x})$ for real s



Shorthand Notation

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

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

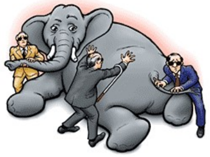
- Note that for real signals $S_{L/2+x}$ = conjugate($S_{L/2-x}$)

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

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

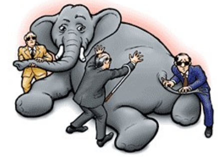


$\mathbf{W^{-1} = W^{H}}$

$$W = \begin{bmatrix} W_L^{0,0} & . & W_L^{L/2,0}. & . & W_L^{L-1,0} \\ W_L^{0,1} & . & W_L^{L/2,1}. & . & W_L^{L-1,1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{0,L-1} & . & W_L^{L/2,L-1} & . & W_L^{L-1,L-1} \end{bmatrix} \qquad W_L^{k,n} = \frac{1}{\sqrt{L}}\exp(j2\pi kn/L)$$

$$W_L^{-k,n} = \frac{1}{\sqrt{L}}\exp(-j2\pi kn/L) \qquad W^H = \begin{bmatrix} W_L^{0,0} & . & W_L^{-0,L/2}. & . & W_L^{-0,L-1} \\ W_L^{-1,0,} & . & W_L^{-1,L/2}. & . & W_L^{-1,L-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_L^{-(L-1),0} & . & W_L^{-(L-1),L/2} & . & W_L^{-(L-1),(L-1)} \end{bmatrix}$$

- The complex exponential basis is orthogonal
  - 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

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

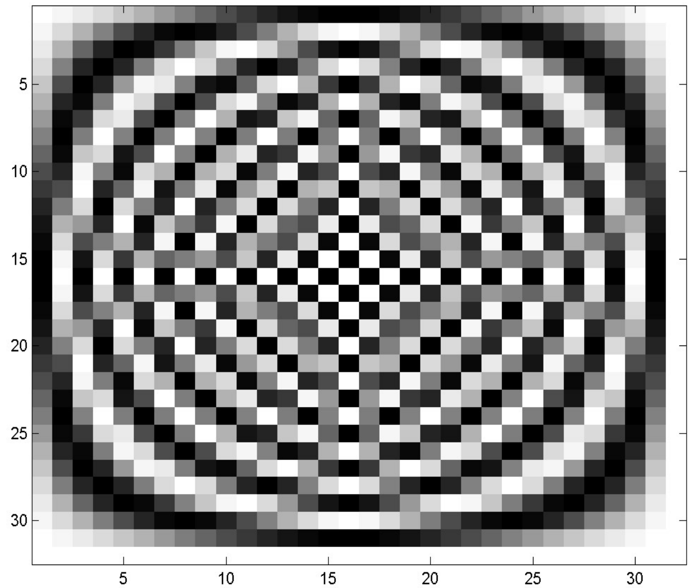
- The matrix to the right is called the "Fourier Matrix"
- The weights (S_0, S_1. . Etc.) are called the Fourier transform

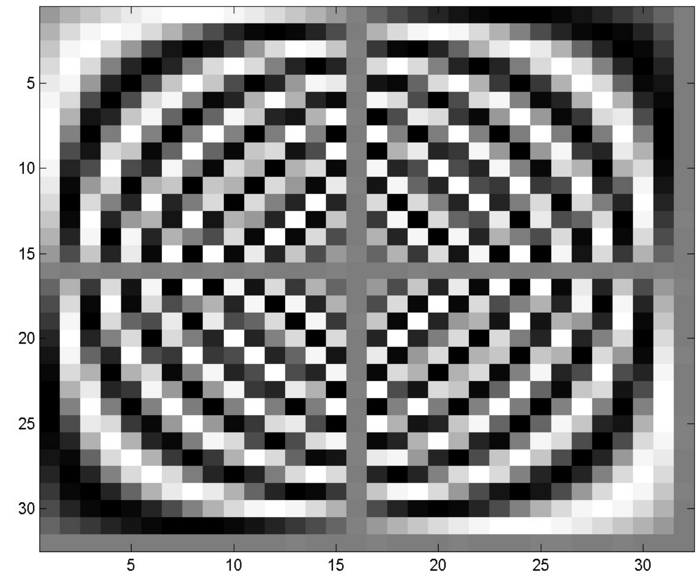
The Inverse Discrete Fourier Transform

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

- 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

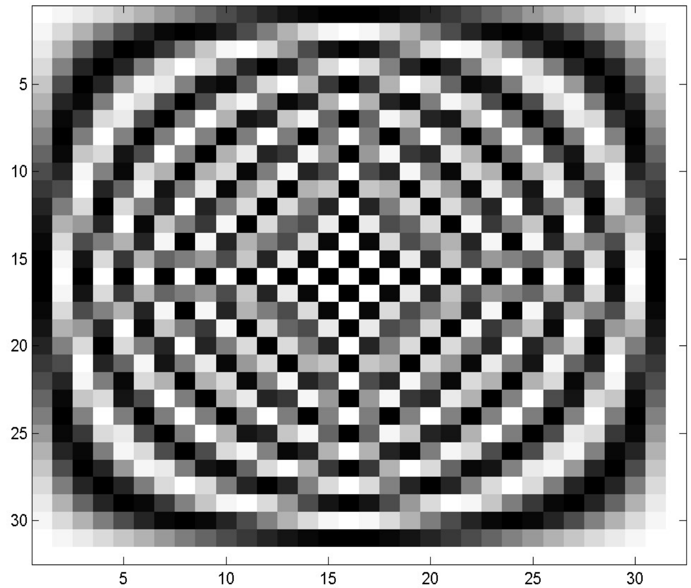
The Fourier Matrix

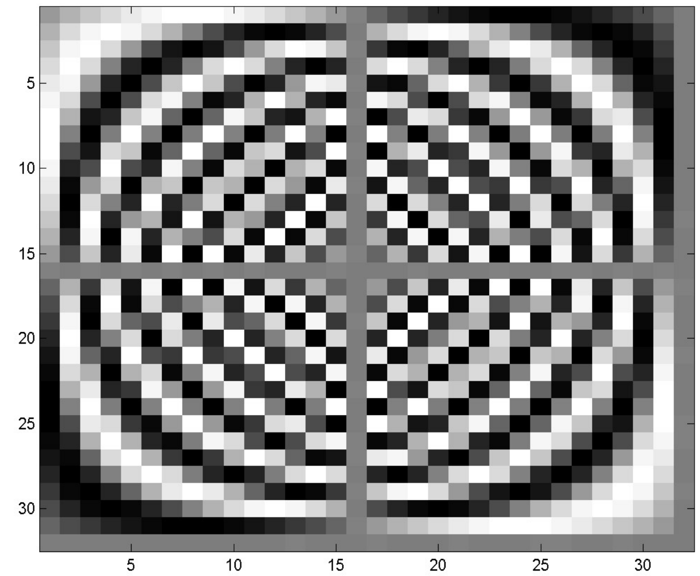




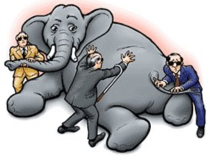
- Left panel: The real part of the Fourier matrix
  - For a 32-point signal
- Right panel: The imaginary part of the Fourier matrix

The FAST Fourier Transform





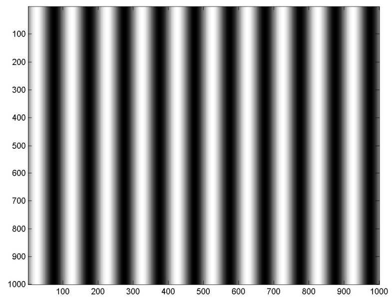
- The outcome of the transformation with the Fourier matrix is the **DISCRETE FOURIER TRANSFORM** (DFT)
- The **FAST Fourier transform** is an algorithm that takes advantage of the symmetry of the matrix to perform the matrix multiplication really fast
- The FFT computes the DFT
  - Is much faster if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

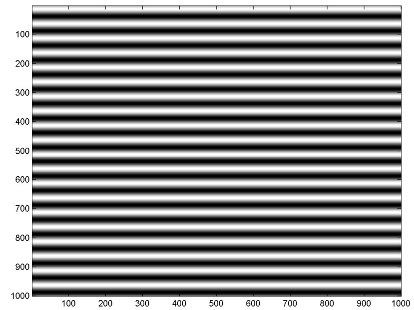


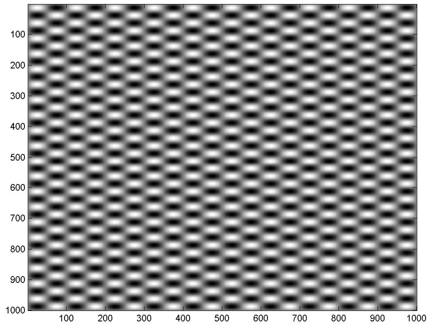
Images

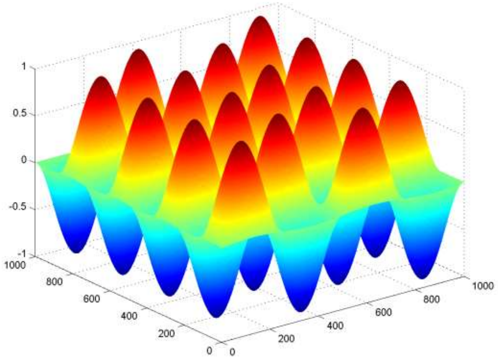
- The complex exponential is two dimensional
  - Has a separate X frequency and Y frequency
    - Would be true even for checker boards!
  - The 2-D complex exponential must be unravelled to form one component of the Fourier matrix
    - For a KxL image, we'd have K*L bases in the matrix

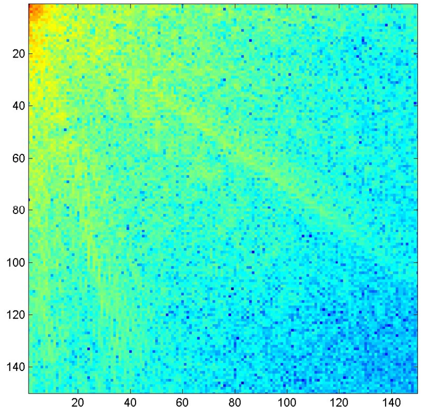
Typical Image Bases





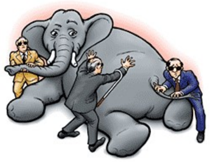






- Only real components of bases shown

DFT: Properties



- The DFT coefficients are complex
  - Have both a magnitude and a phase

$$S_k = |S_k| \exp(-j\angle S_k)$$

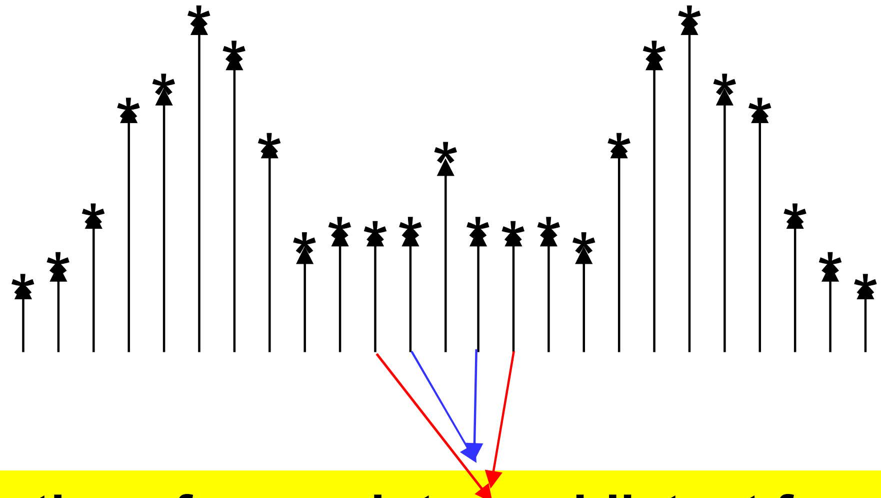
- Simple linear algebra tells us that
  - DFT(A + B) = DFT(A) + DFT(B)
  - The DFT of the sum of two signals is the DFT of their sum

- A horribly common approximation in sound processing
  - Magnitude(DFT(A+B)) = Magnitude(DFT(A)) + Magnitude(DFT(B))
  - Utterly wrong
  - Absurdly useful

Index

- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

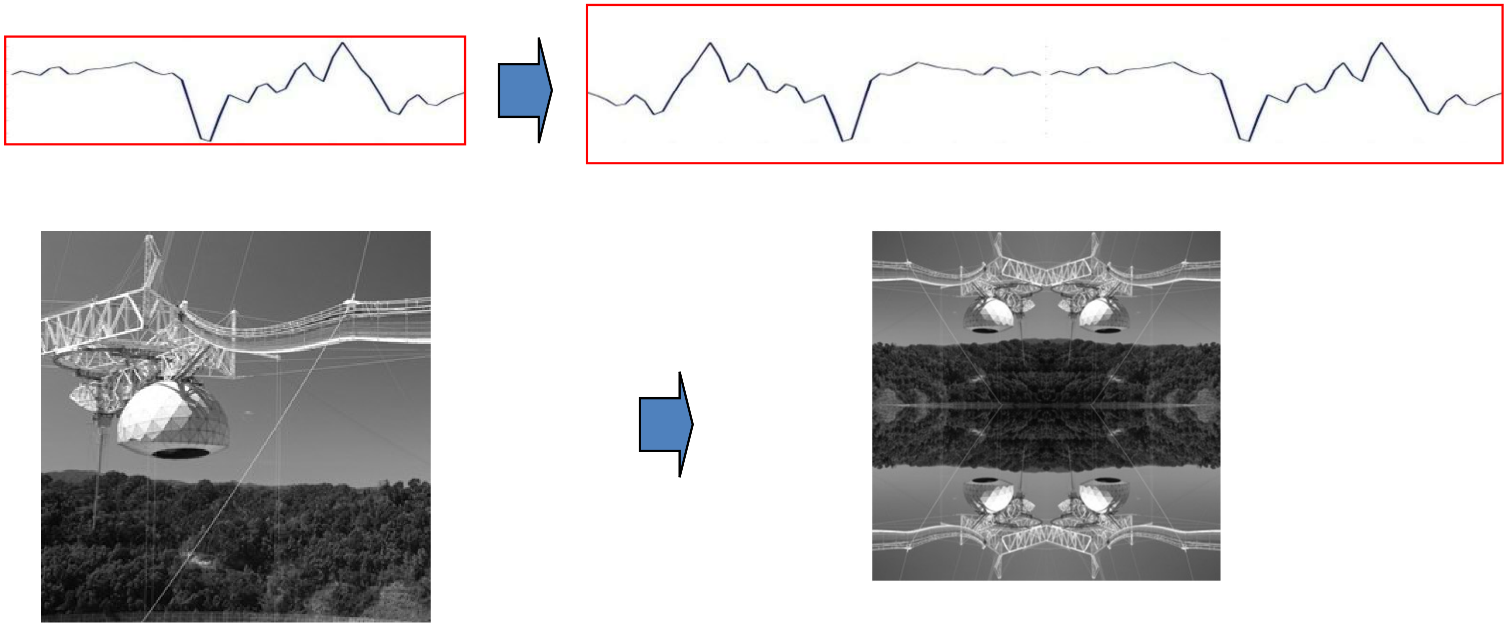
Symmetric signals



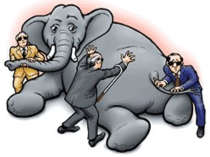
Contributions from points equidistant from L/2 combine to cancel out imaginary terms

- If a signal is (conjugate) symmetric around L/2, the Fourier coefficients are real!
  - $A(L/2-k) * \exp(-j *f*(L/2-k)) + A(L/2+k) * \exp(-j*f*(L/2+k))$ is always real if
    $A(L/2-k) = \text{conjugate}(A(L/2+k))$
  - 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



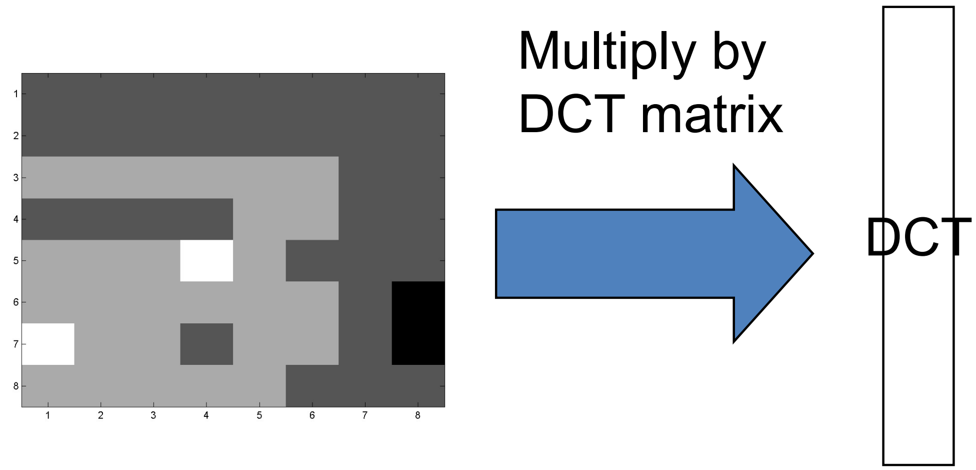
DCT

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

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

Images and DCT



- 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 Sound and Images

- "Deterministic" representations of audio time series and image data..

Aside: some tricks to computing Fourier transforms

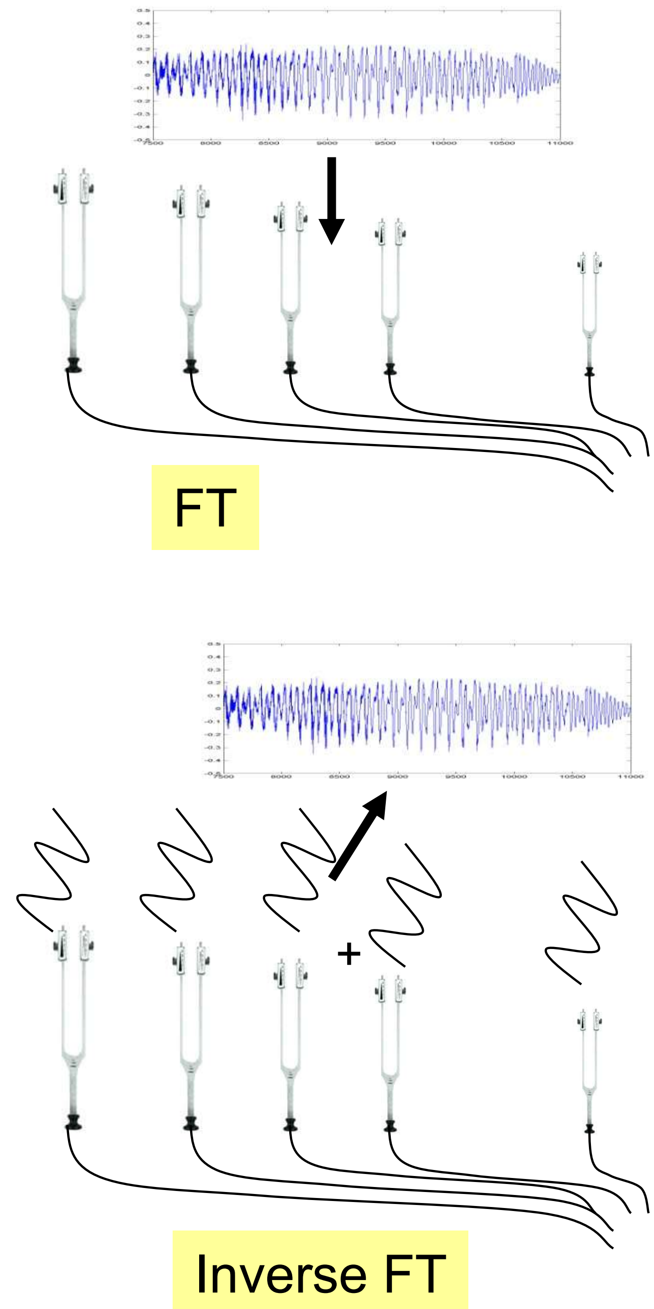
- Direct computation of the Fourier transform can result in poor representations
- Boundary effects can cause error
  - Solution : Windowing
- The size of the signal can introduce inefficiency
  - Solution: Zero padding

Index

- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

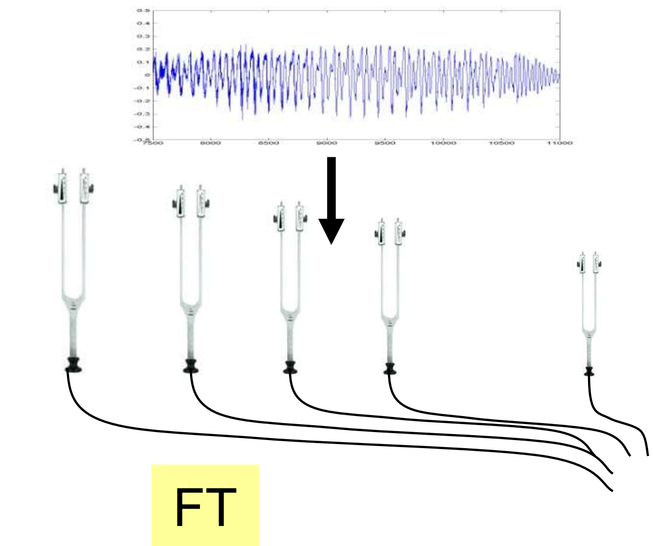
Sound: A thought experiment

- Analysis: Analyze the sound using a bank of tuning forks
- Transduce the vibrations and store / transmit them
- Synthesis: Activate tuning forks with the transduced signal
- *What do we get?*

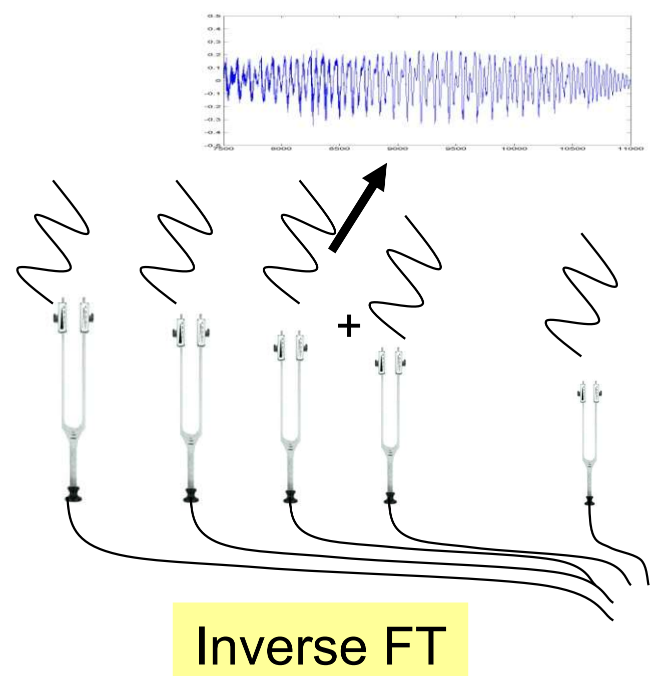


The Fourier Transform and Perception: Sound

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

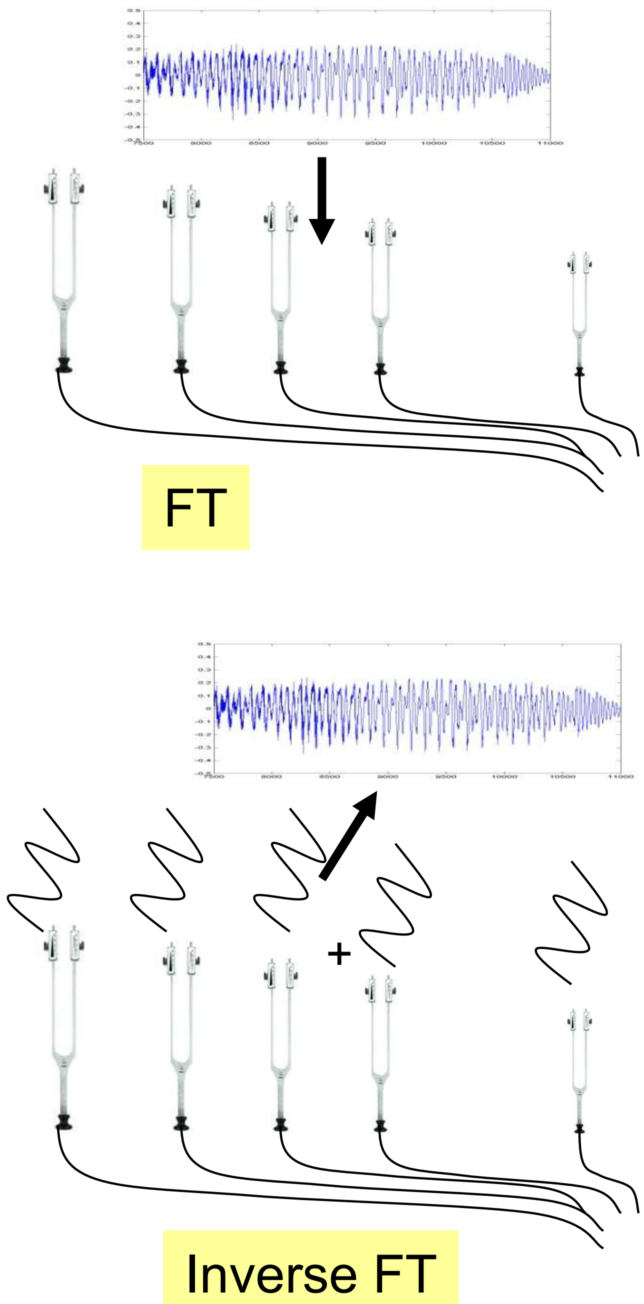


- Our ear *has* a bank of tuning forks
- The output of the Fourier transform is perceptually very meaningful

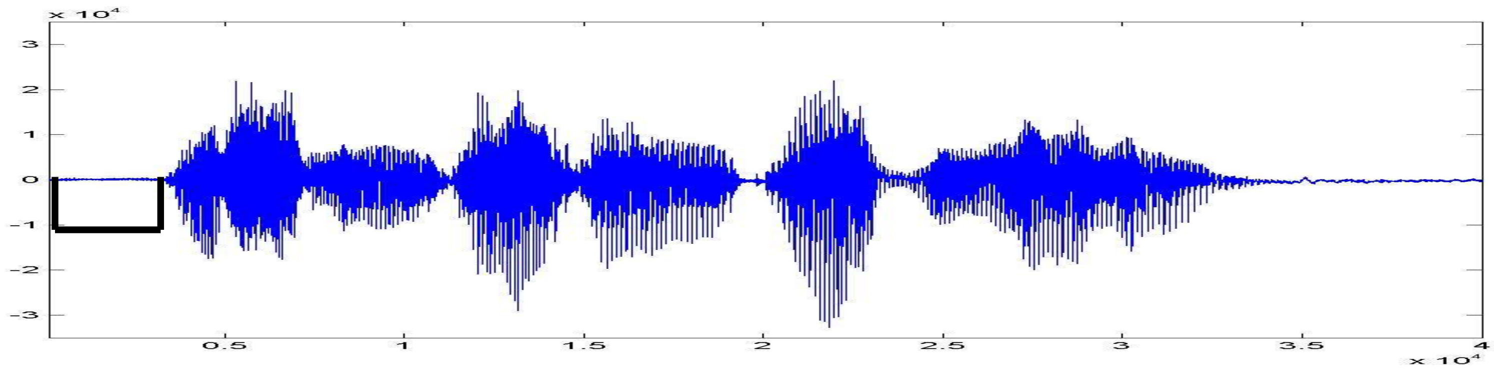


The Fourier Transform and Perception: Sound

- Processing Sound:
- Analyze the sound using a bank of tuning forks
- *Sample* the transduced output of the turning forks at periodic intervals

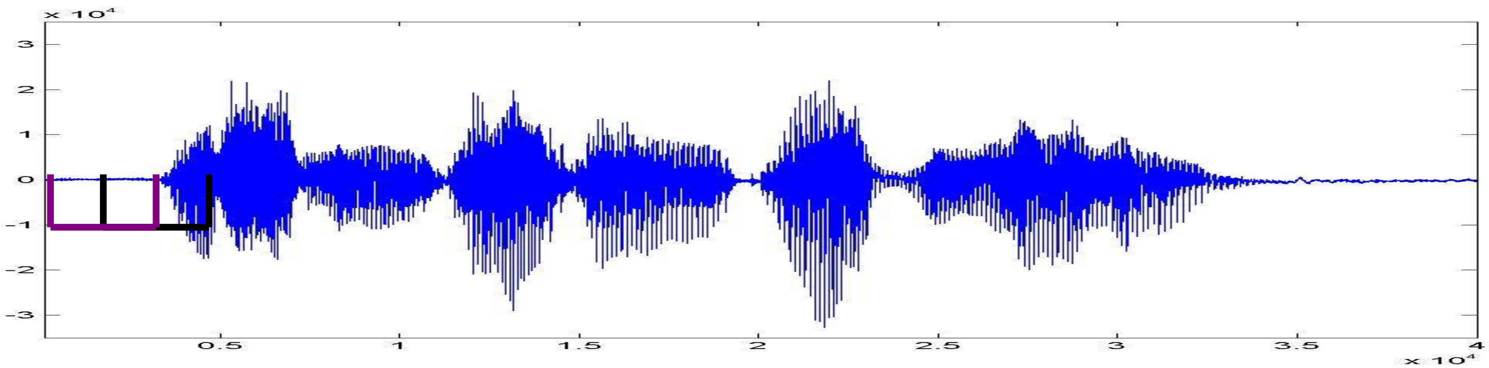


Sound parameterization



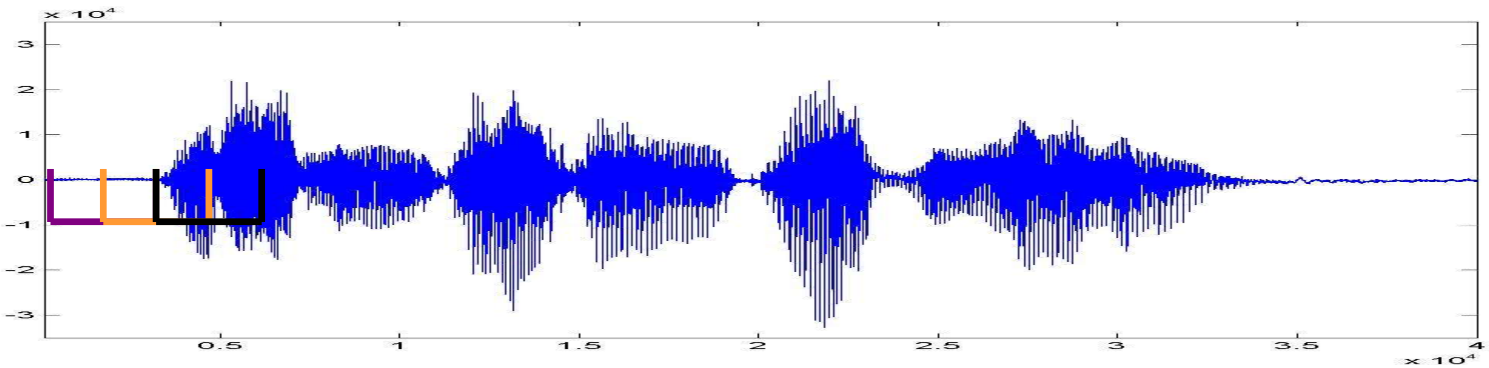
- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are “stationary” only very briefly

Sound parameterization



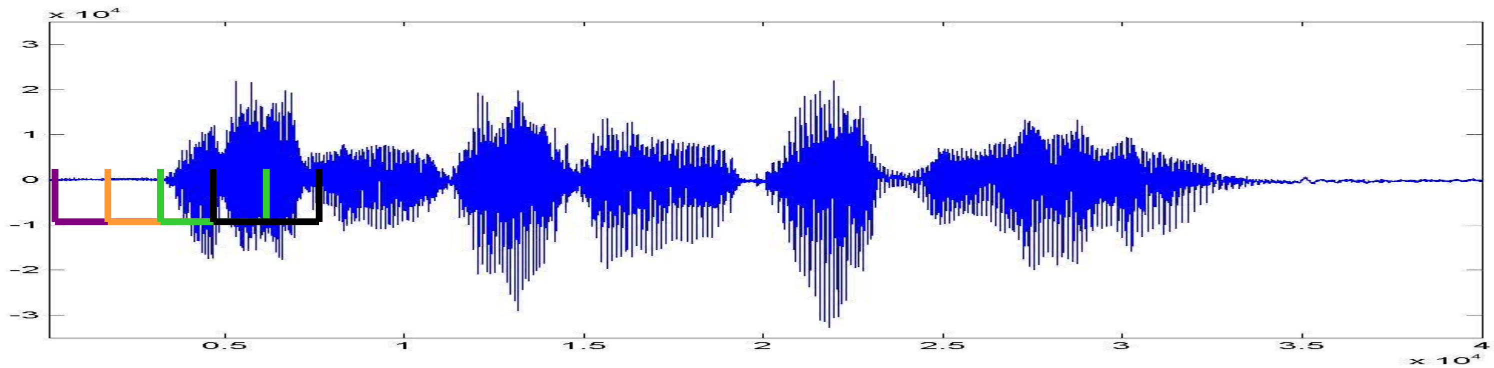
- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are “stationary” only very briefly
- Adjacent segments overlap by 15-48 ms

Sound parameterization



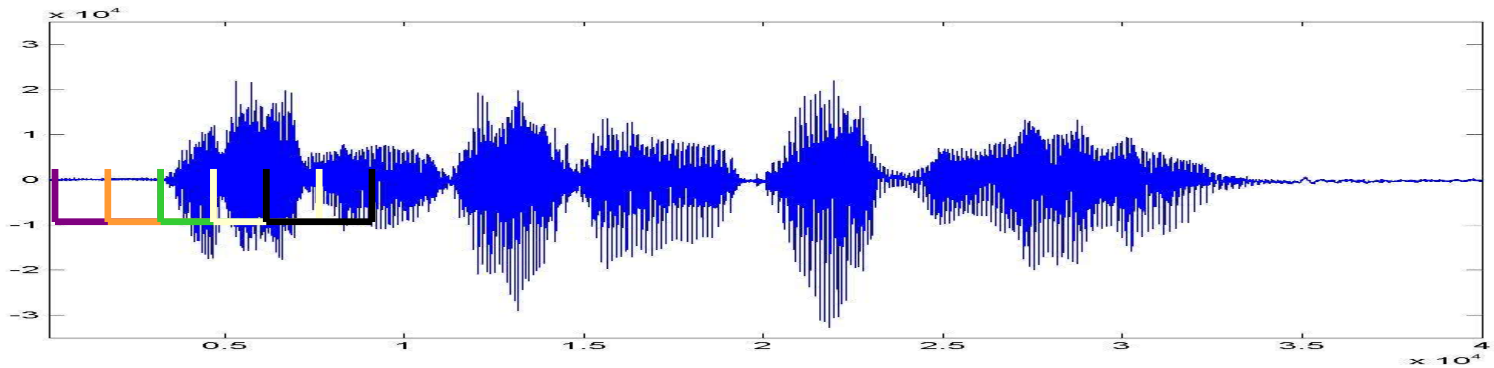
- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are "stationary" only very briefly
- Adjacent segments overlap by 15-48 ms

Sound parameterization



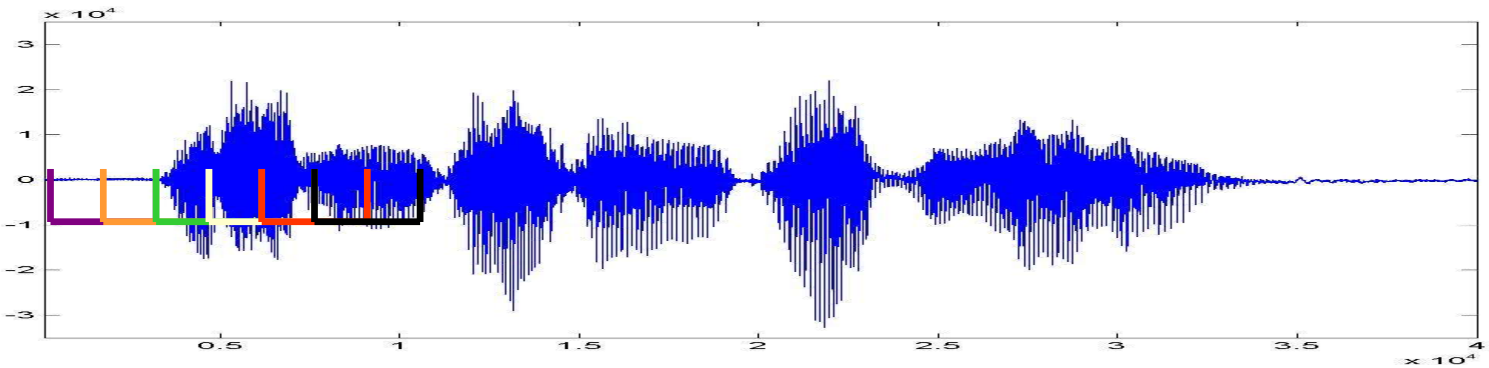
- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are “stationary” only very briefly
- Adjacent segments overlap by 15-48 ms

Sound parameterization



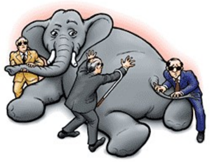
- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are "stationary" only very briefly
- Adjacent segments overlap by 15-48 ms

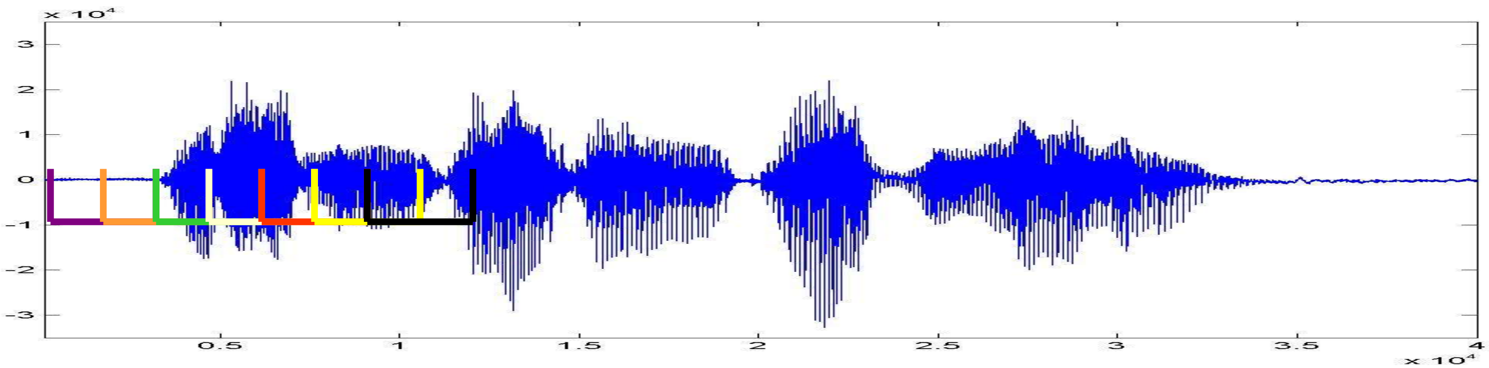
Sound parameterization



- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are “stationary” only very briefly
- Adjacent segments overlap by 15-48 ms

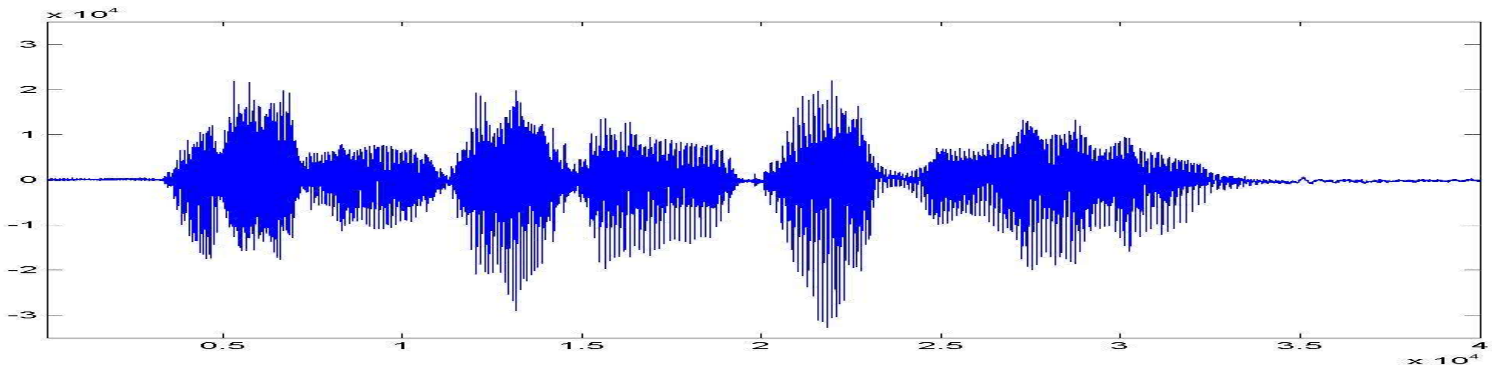
Sound parameterization

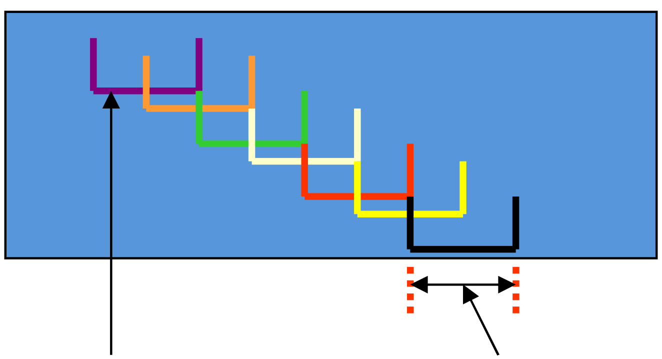




- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are "stationary" only very briefly
- Adjacent segments overlap by 15-48 ms

Sound parameterization



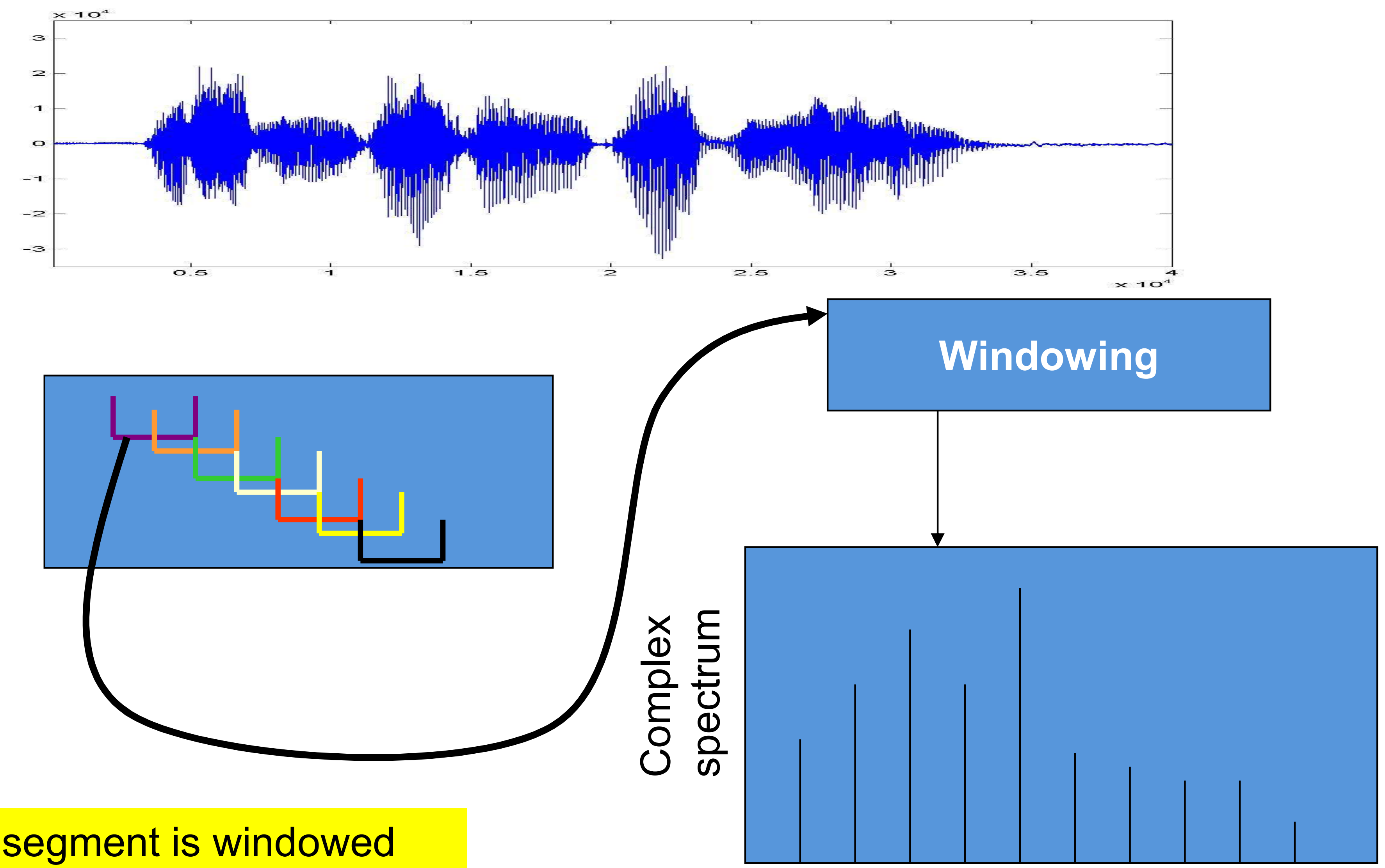


Segments shift every 10-16 milliseconds

Each segment is typically 25-64 milliseconds wide

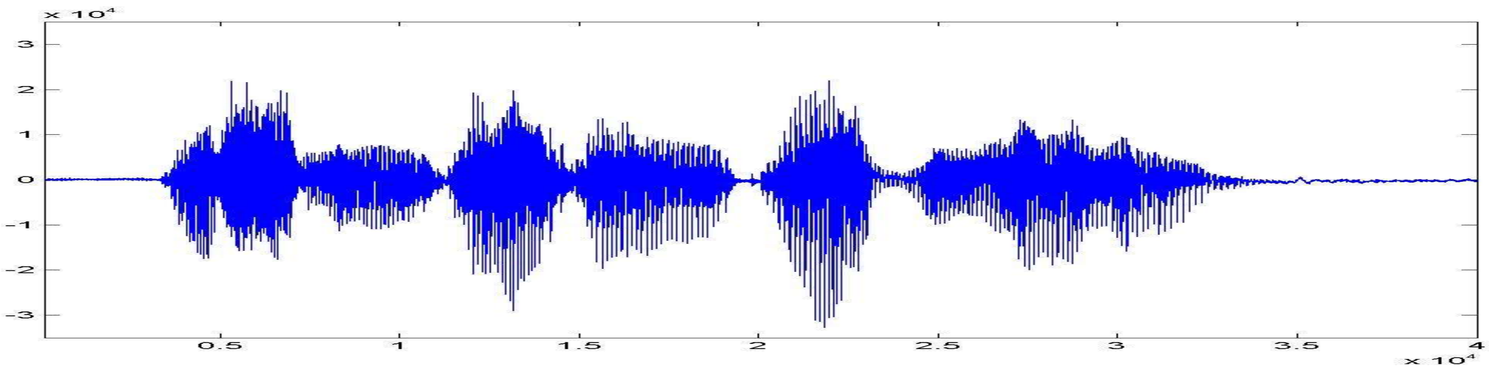
Audio signals typically do not change significantly within this short time interval

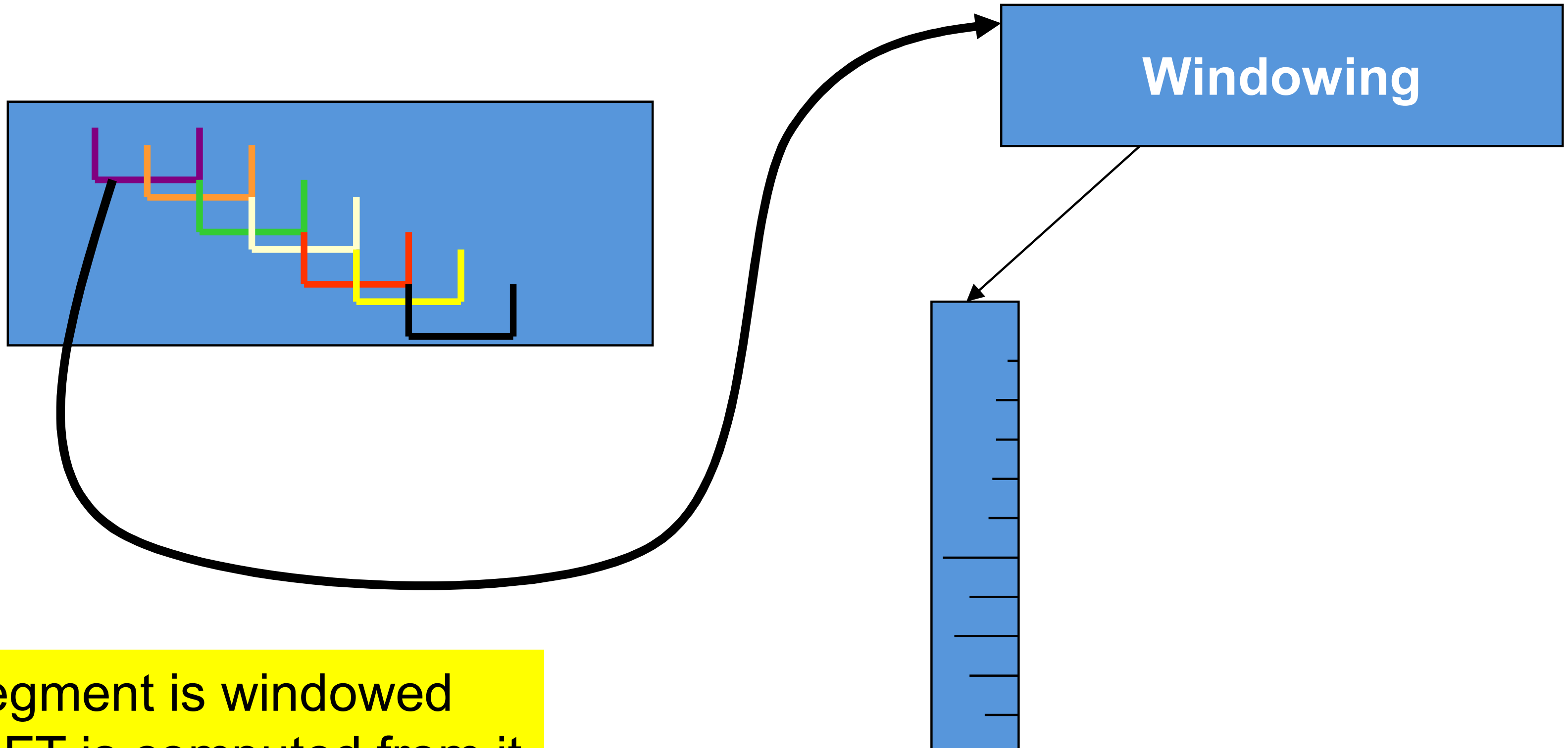
Sound parameterization



Each segment is windowed and a DFT is computed from it

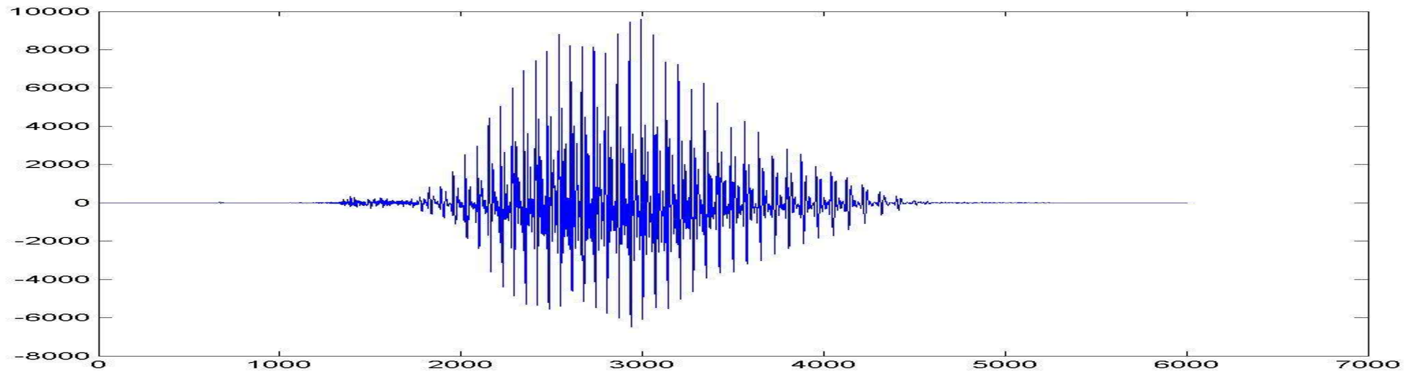
Sound parameterization

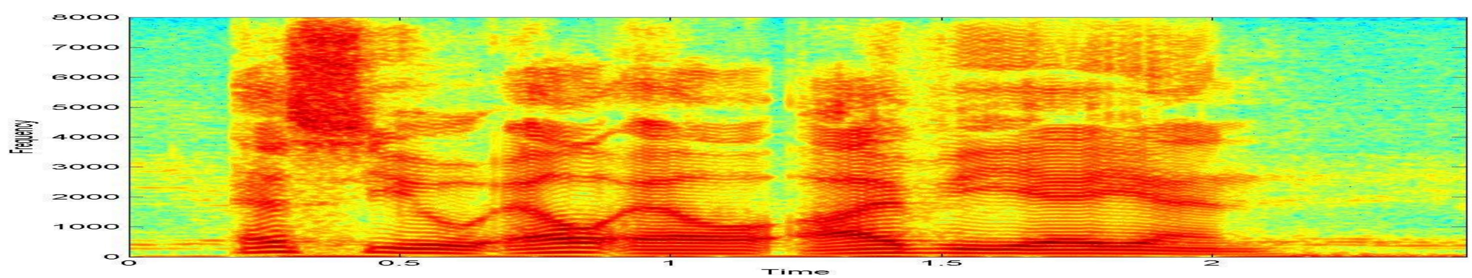




Each segment is windowed
and a DFT is computed from it

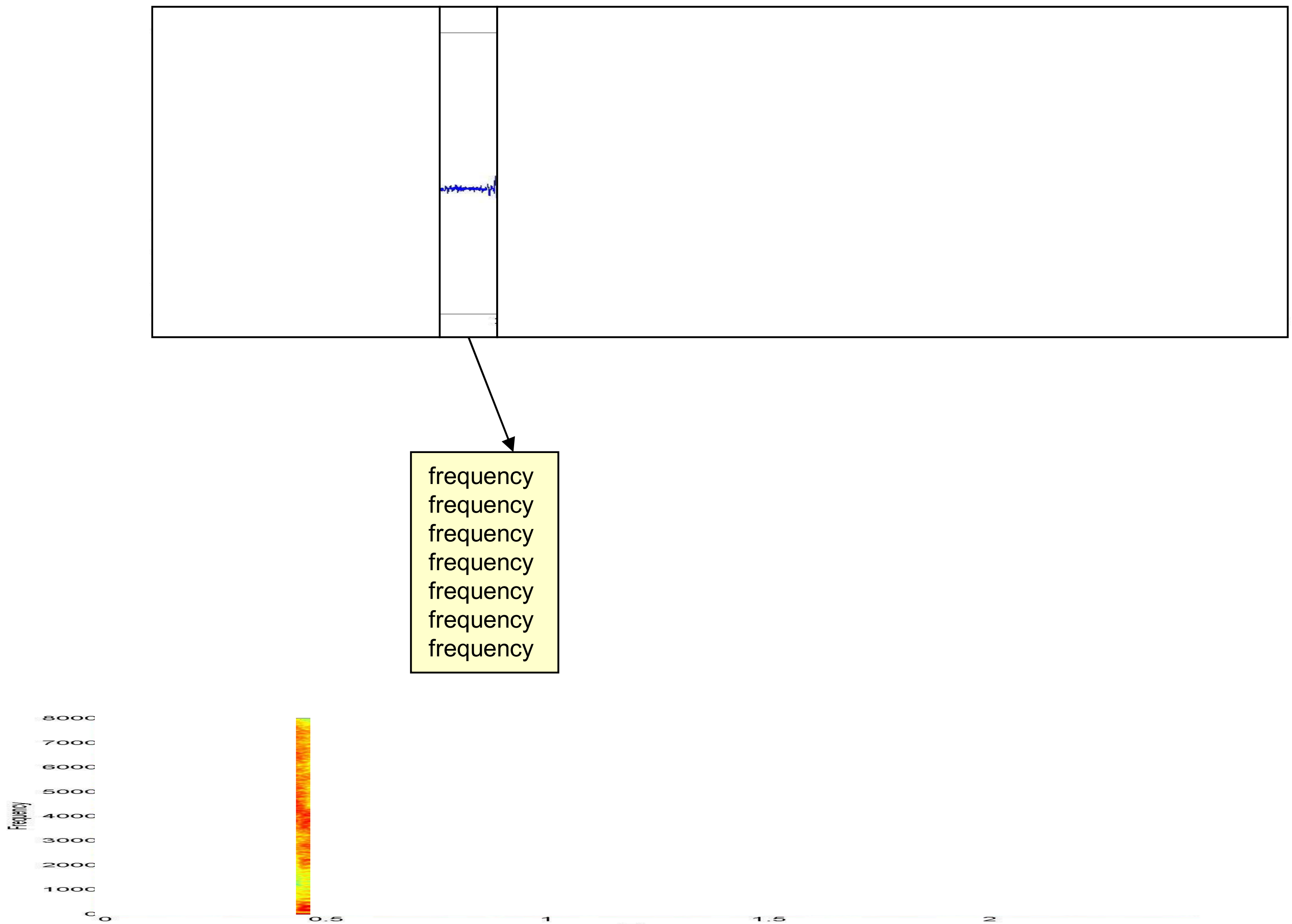
Computing a Spectrogram





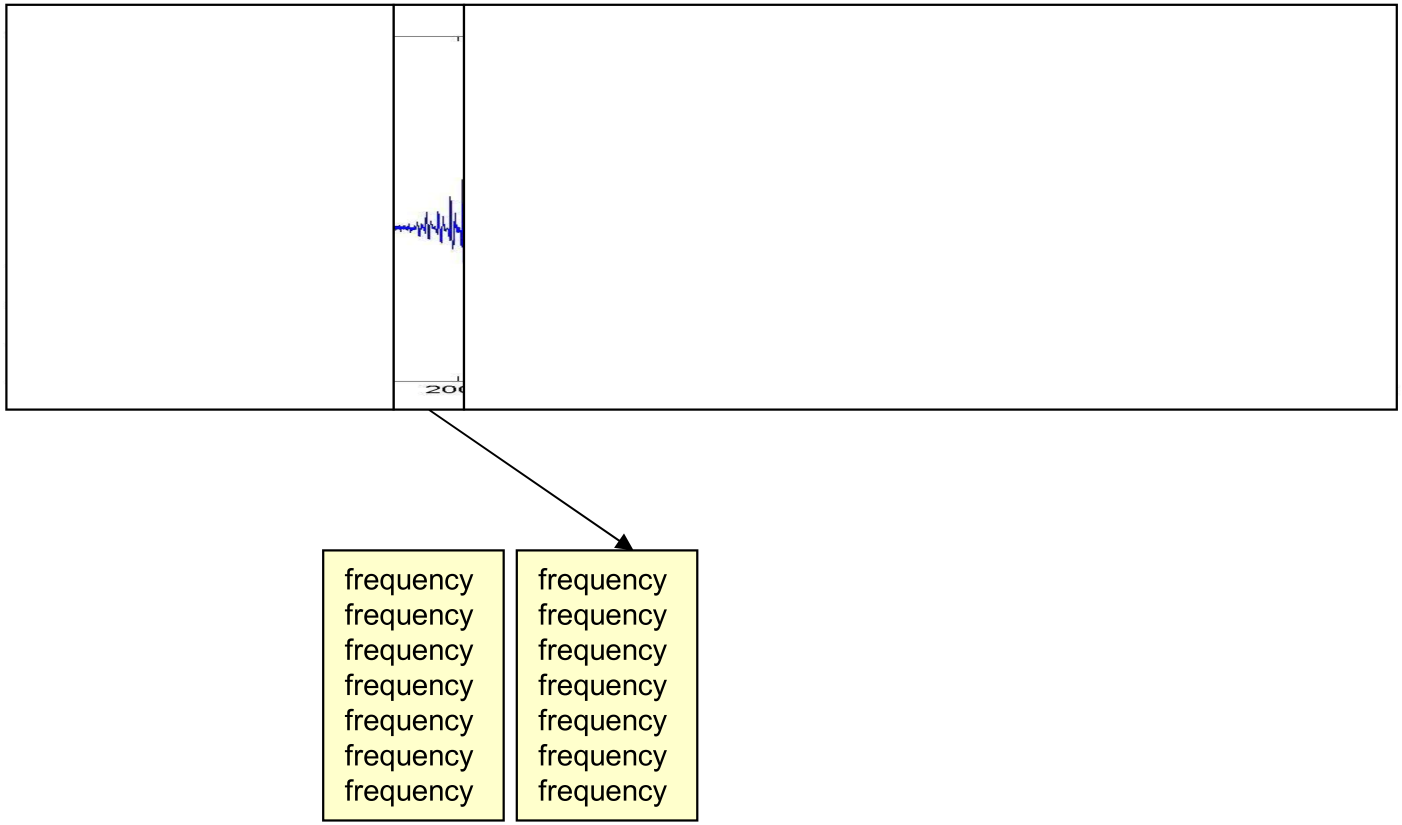
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



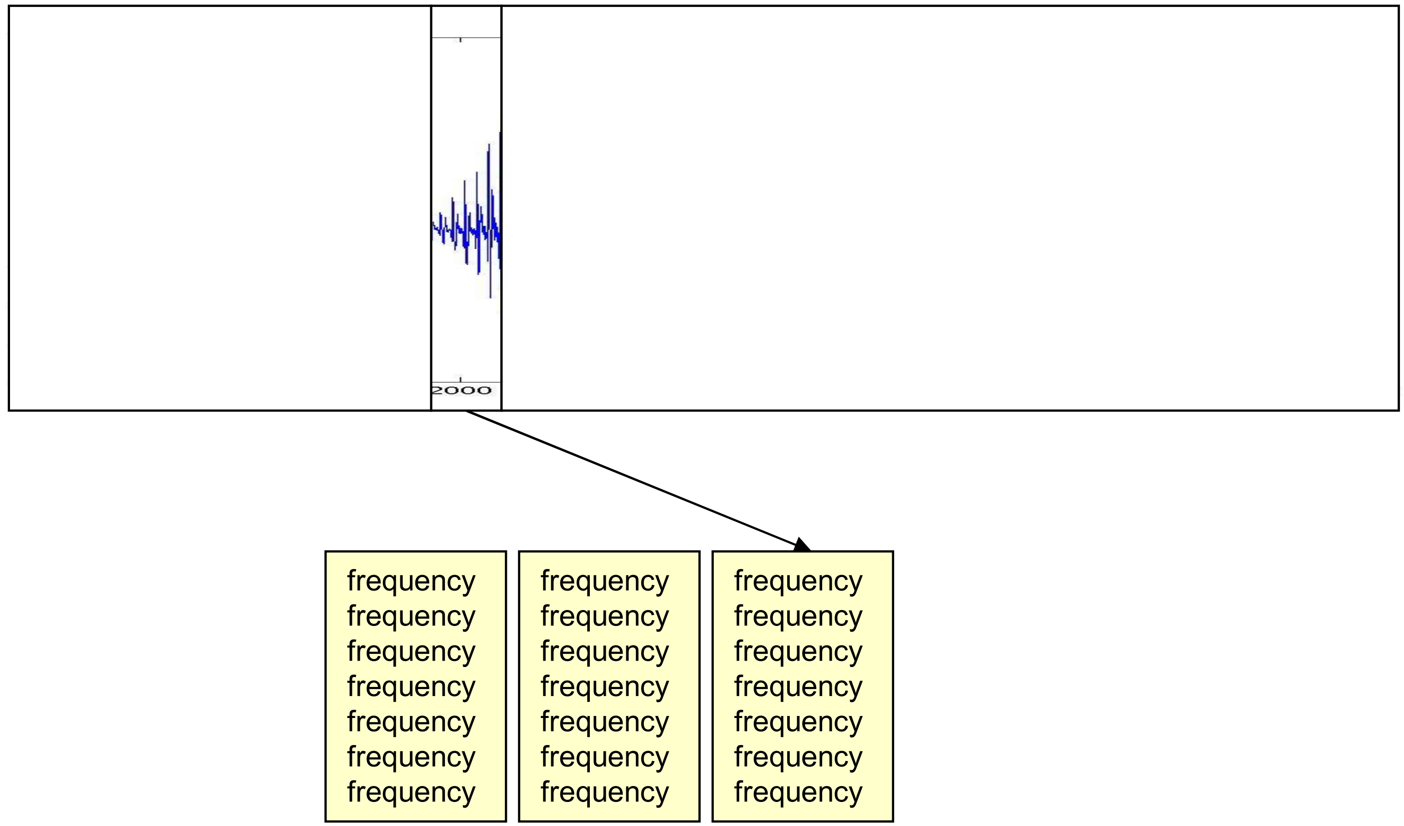
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



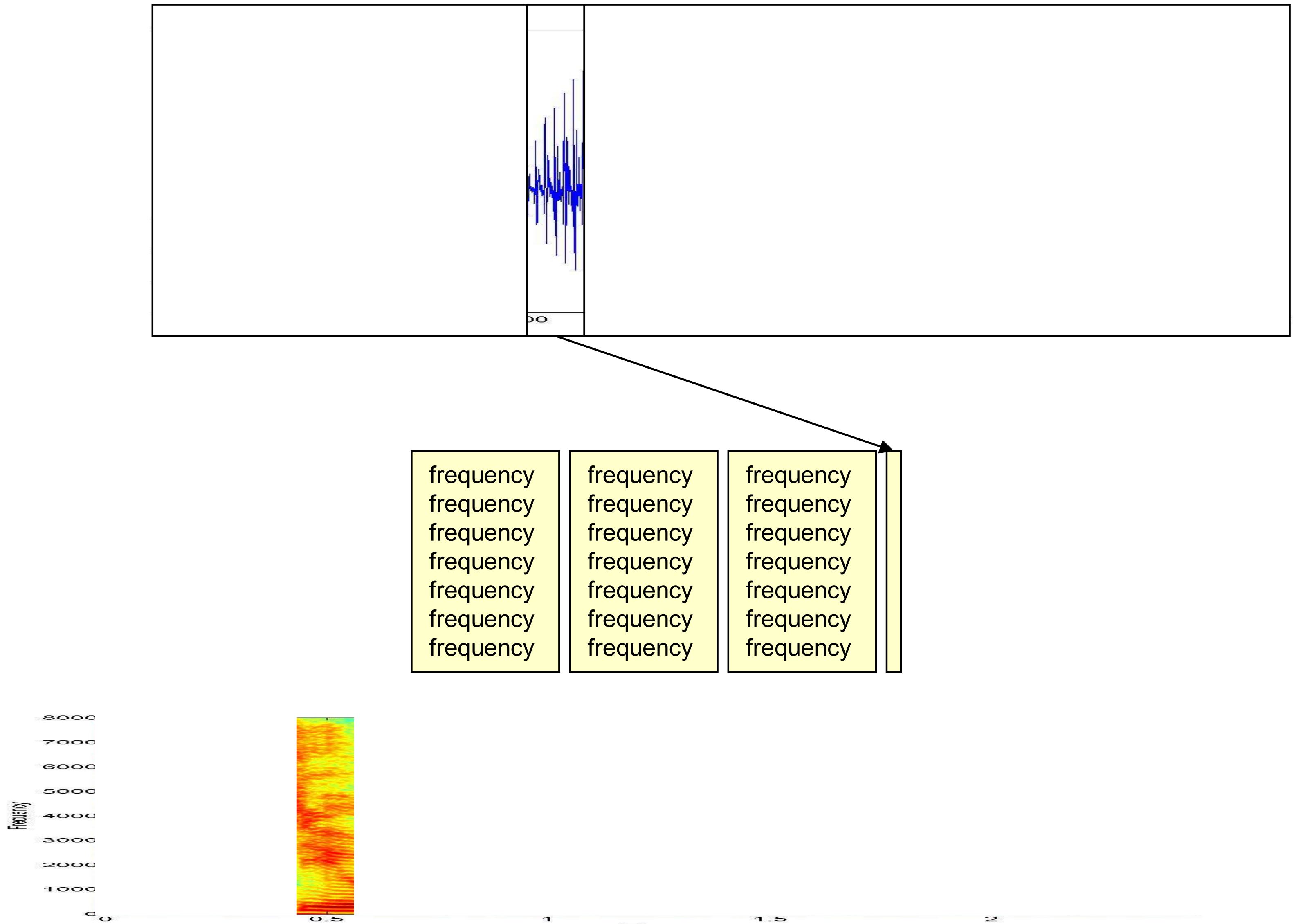
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



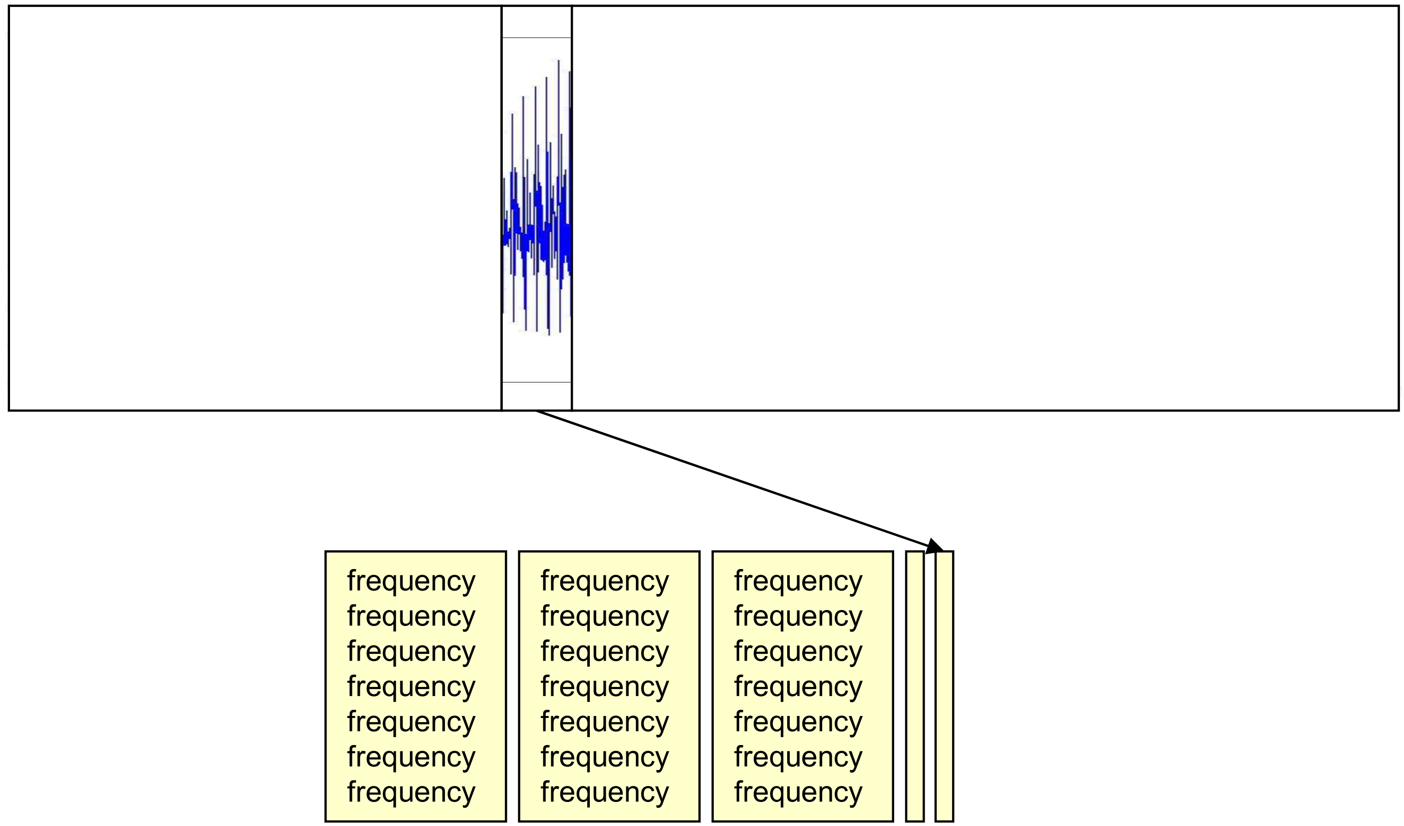
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



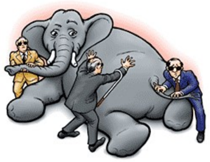
Compute Fourier Spectra of segments of audio and stack them side-by-side

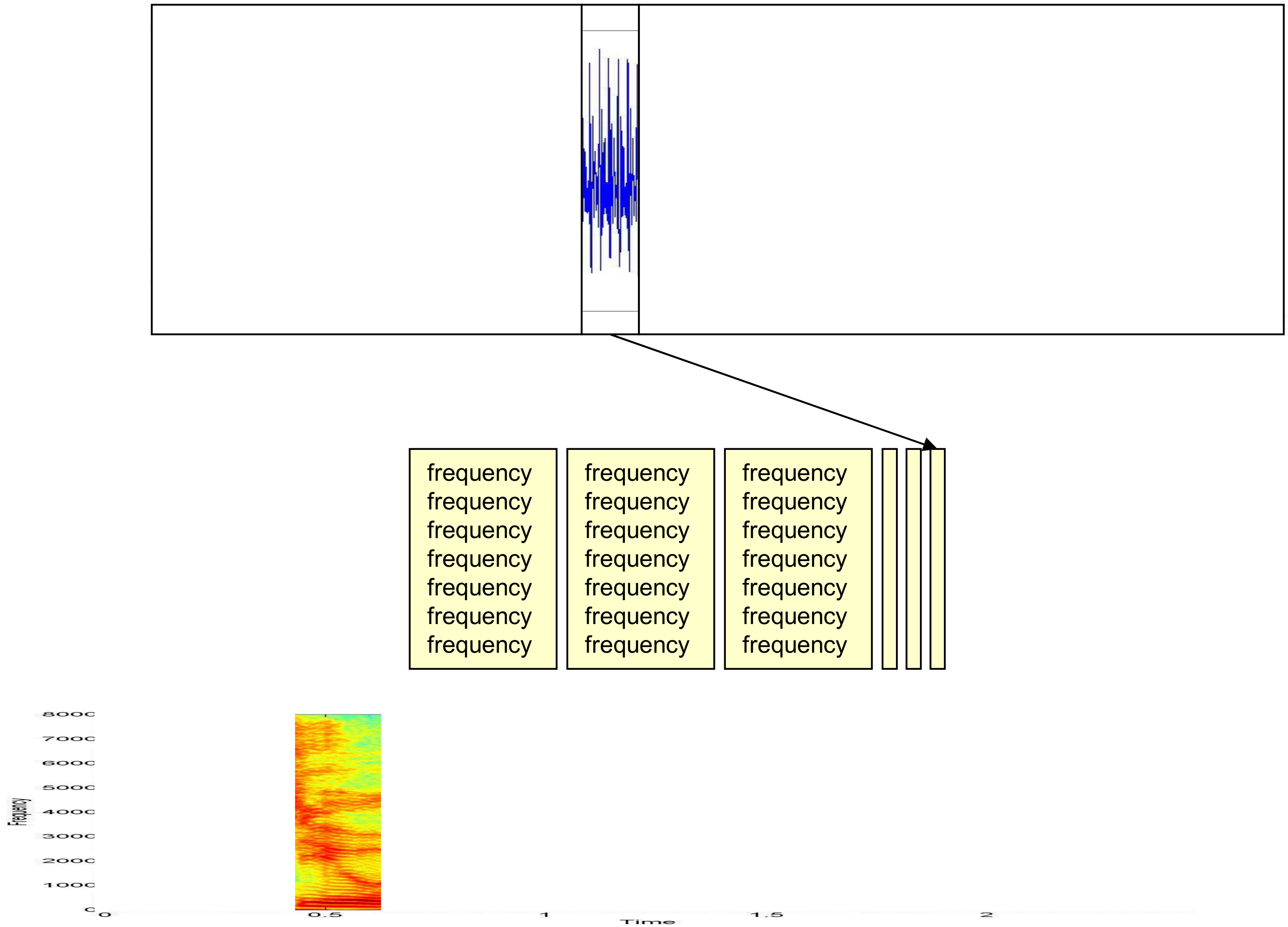
Computing a Spectrogram



Compute Fourier Spectra of segments of audio and stack them side-by-side

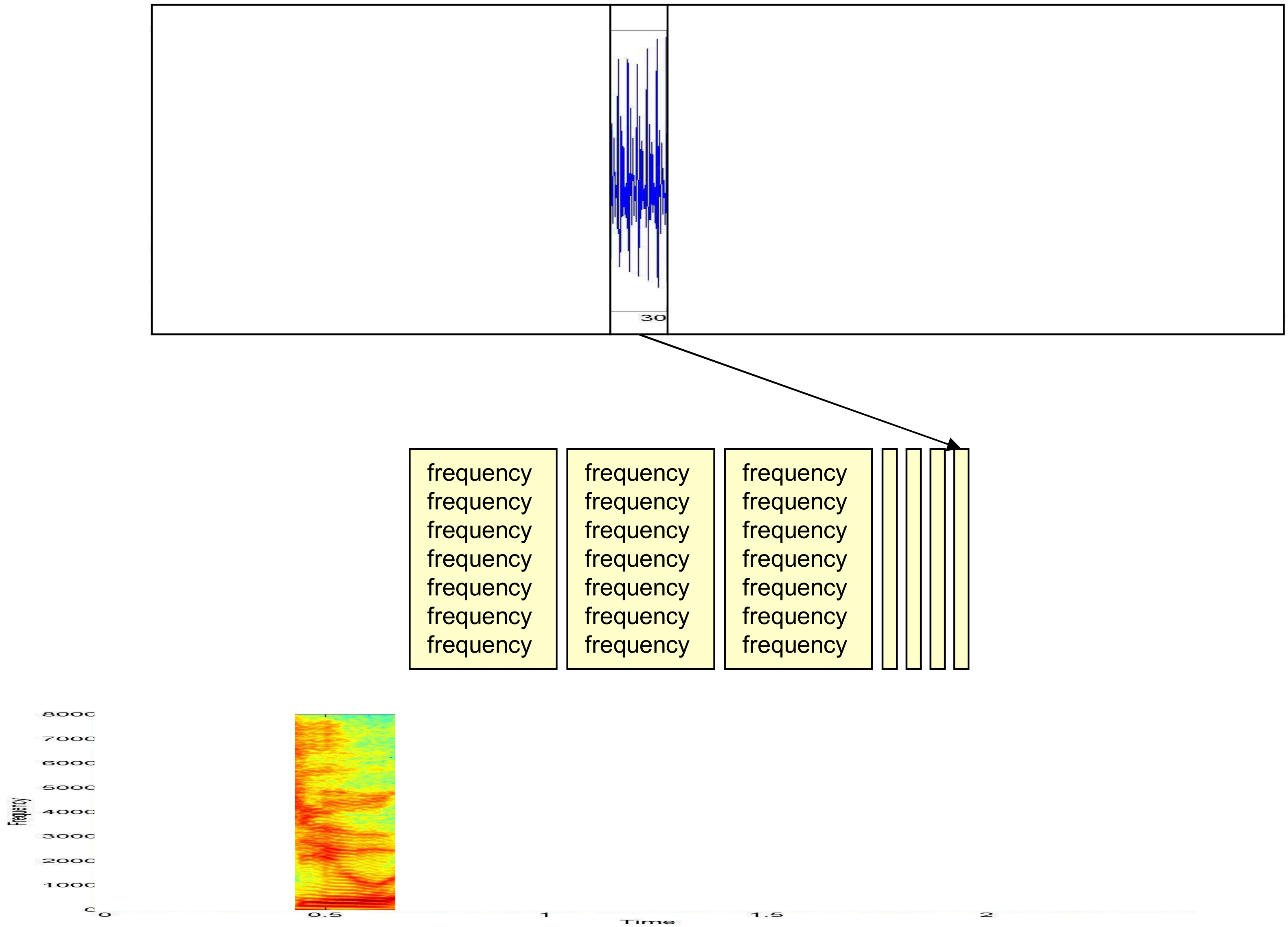
Computing a Spectrogram





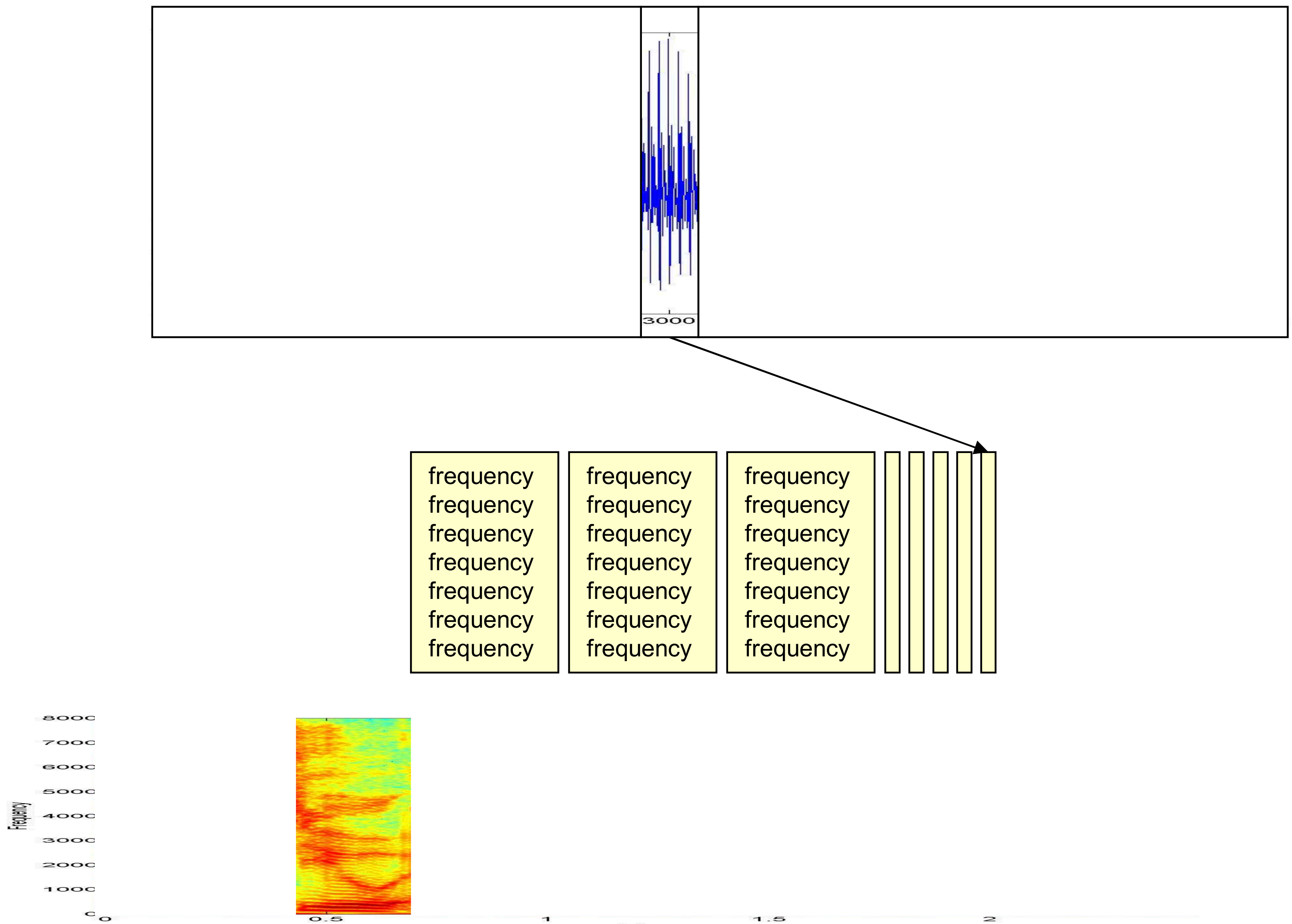
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



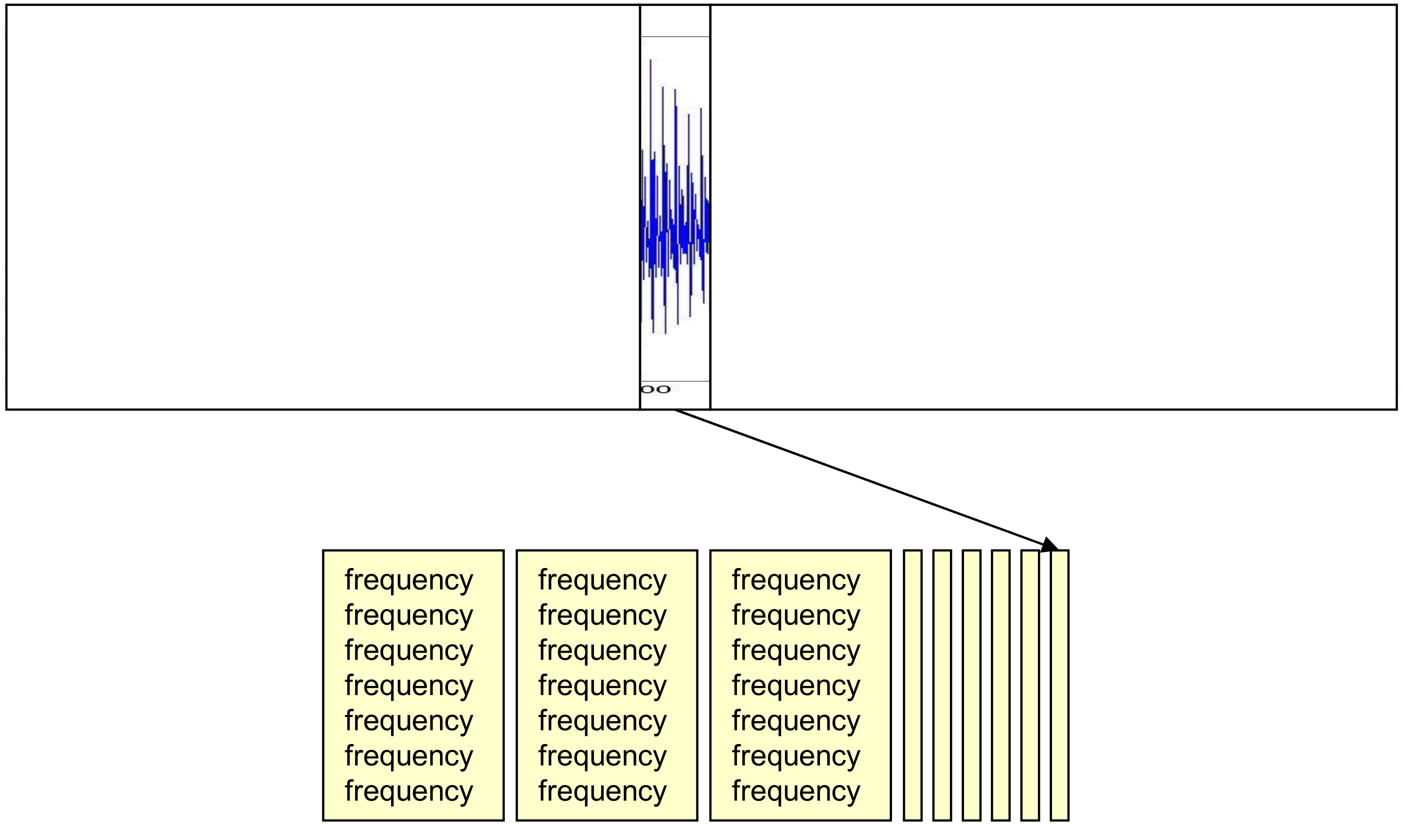
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



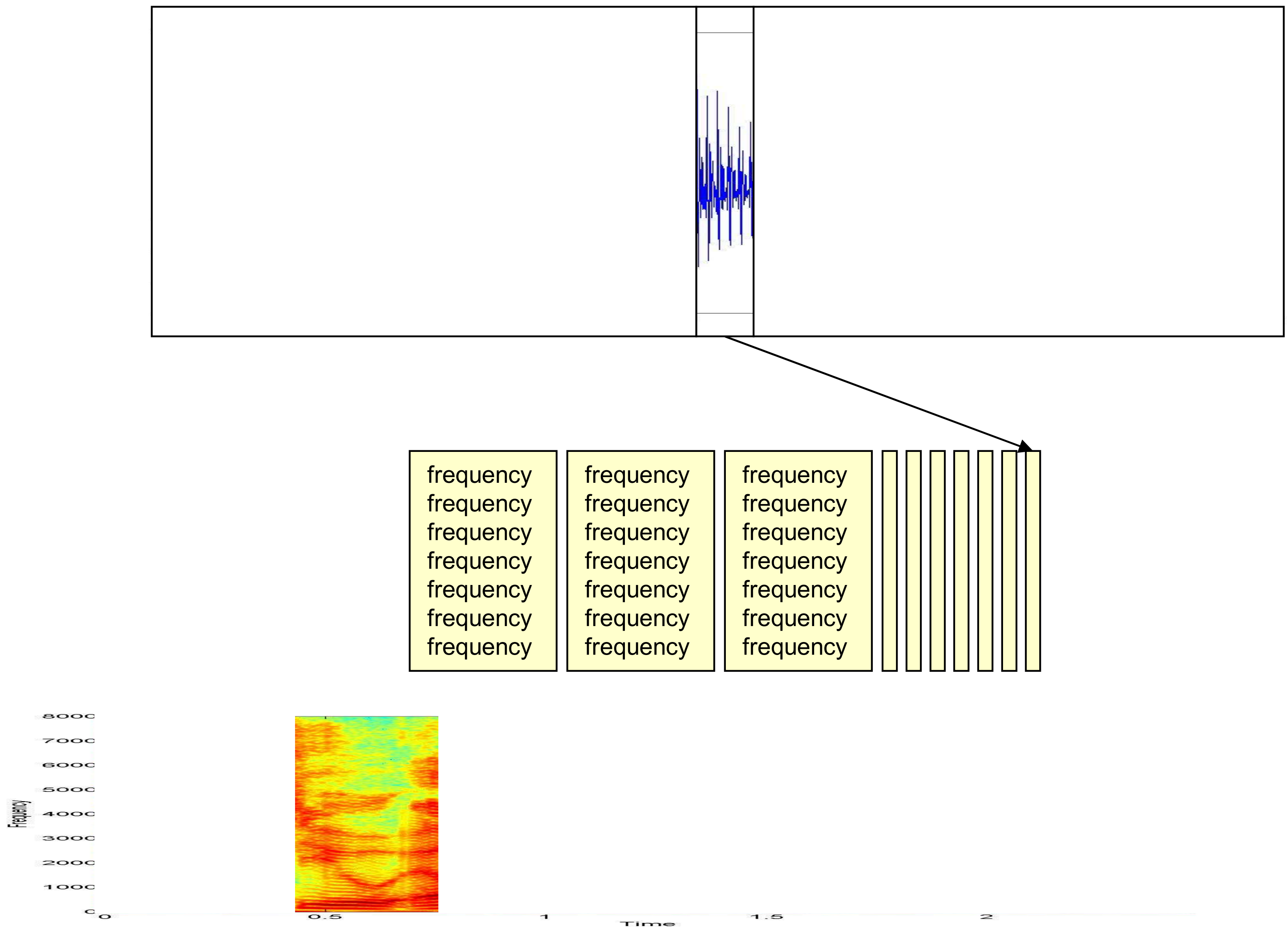
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



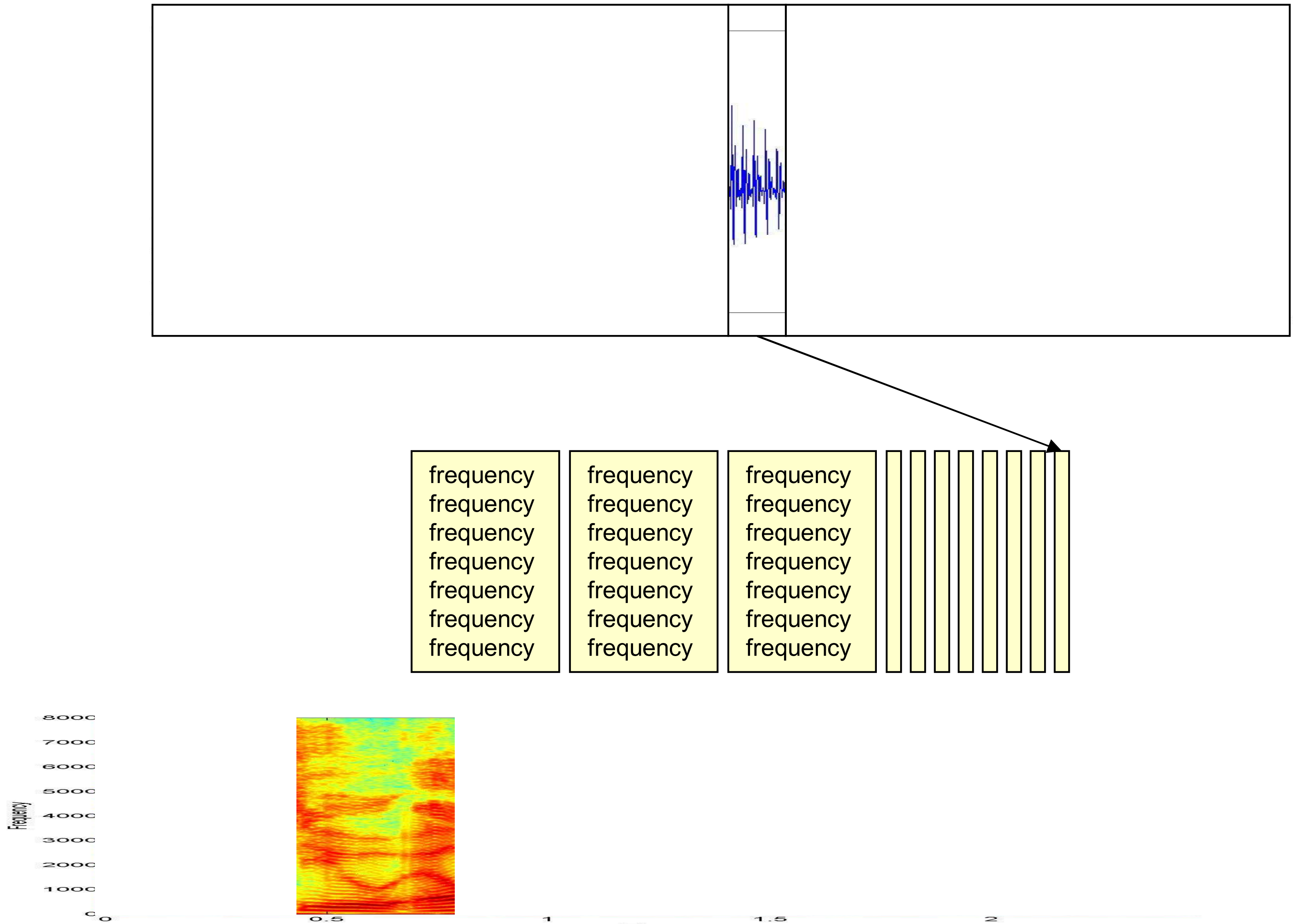
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



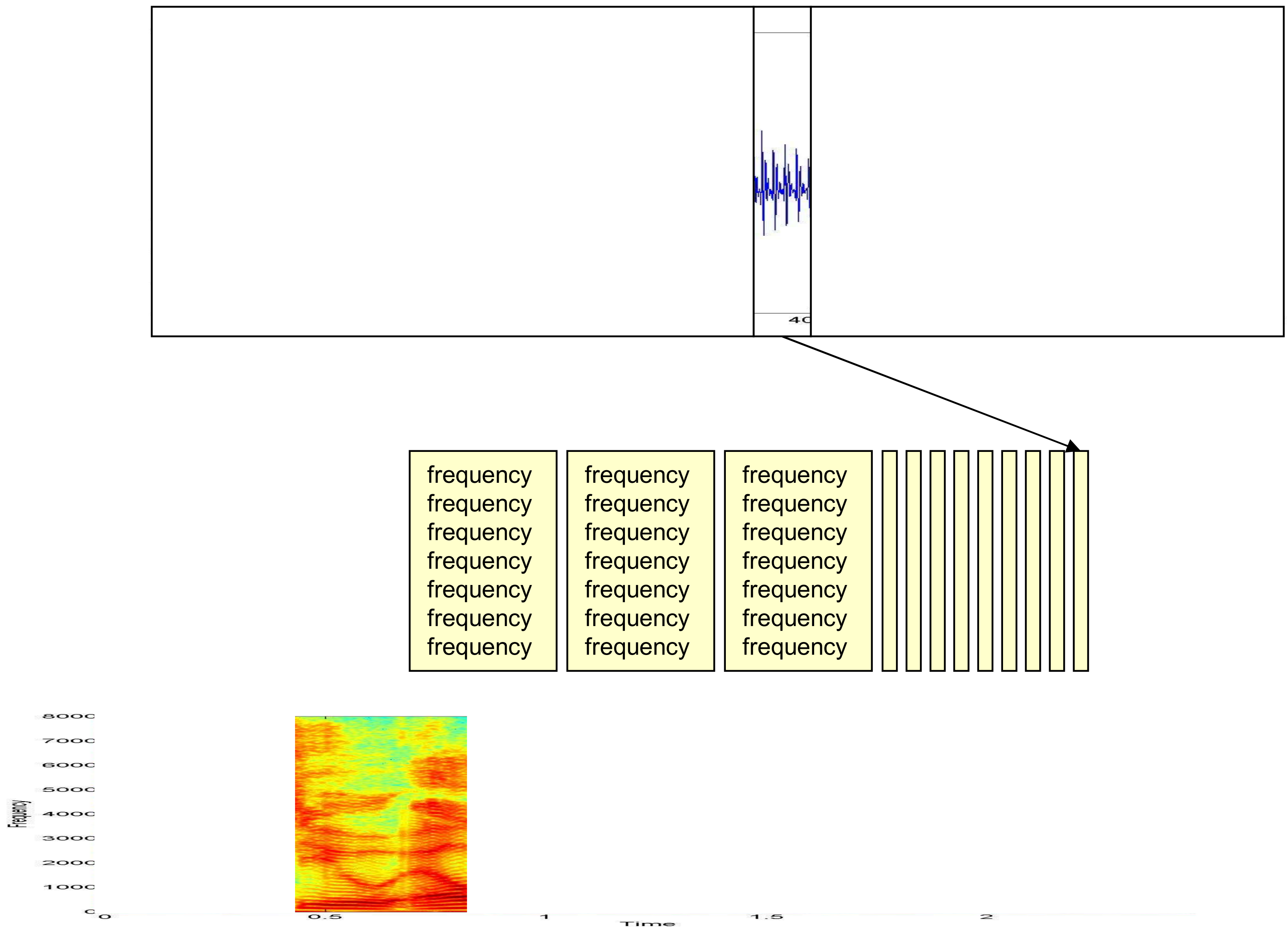
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



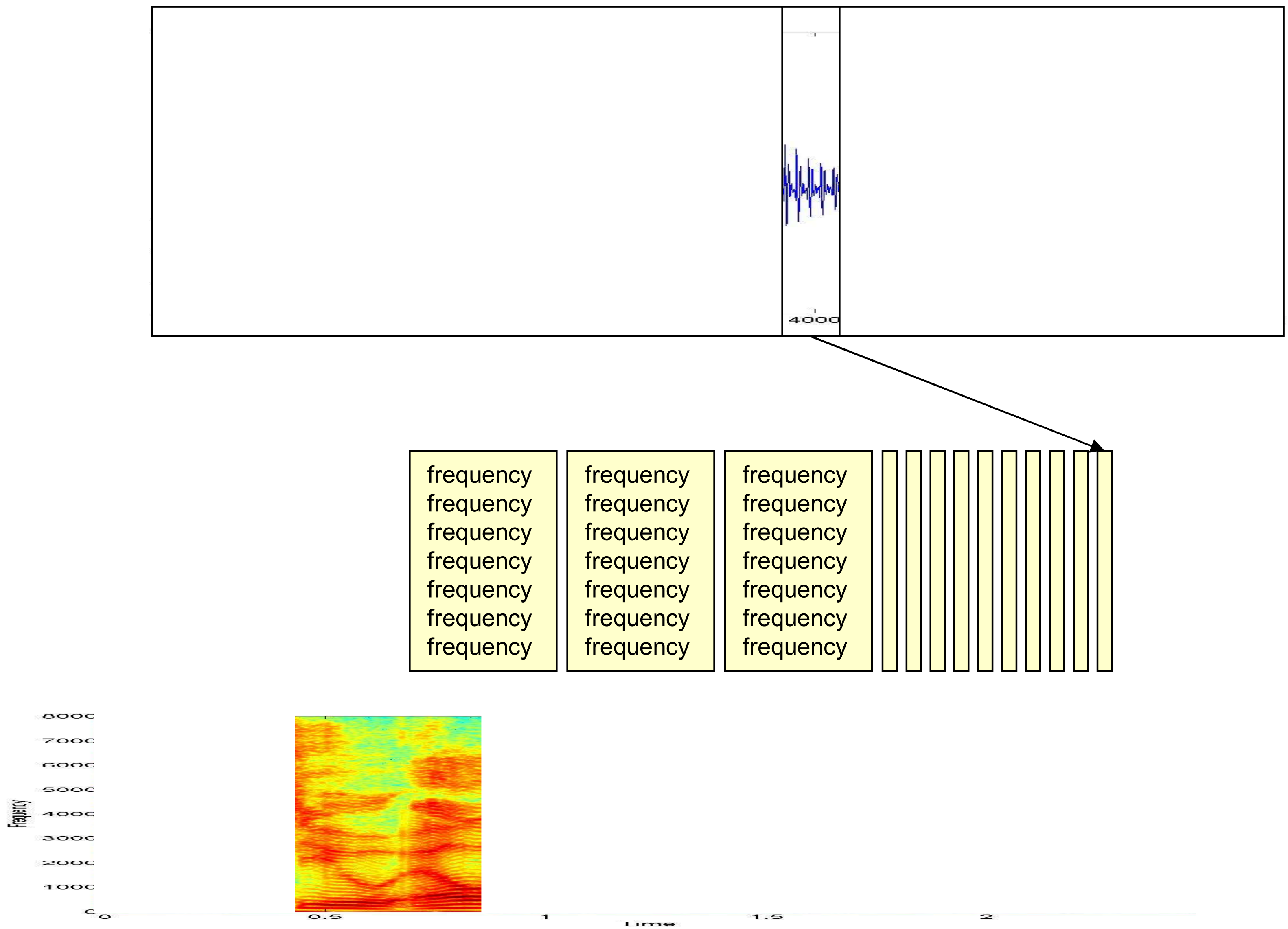
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



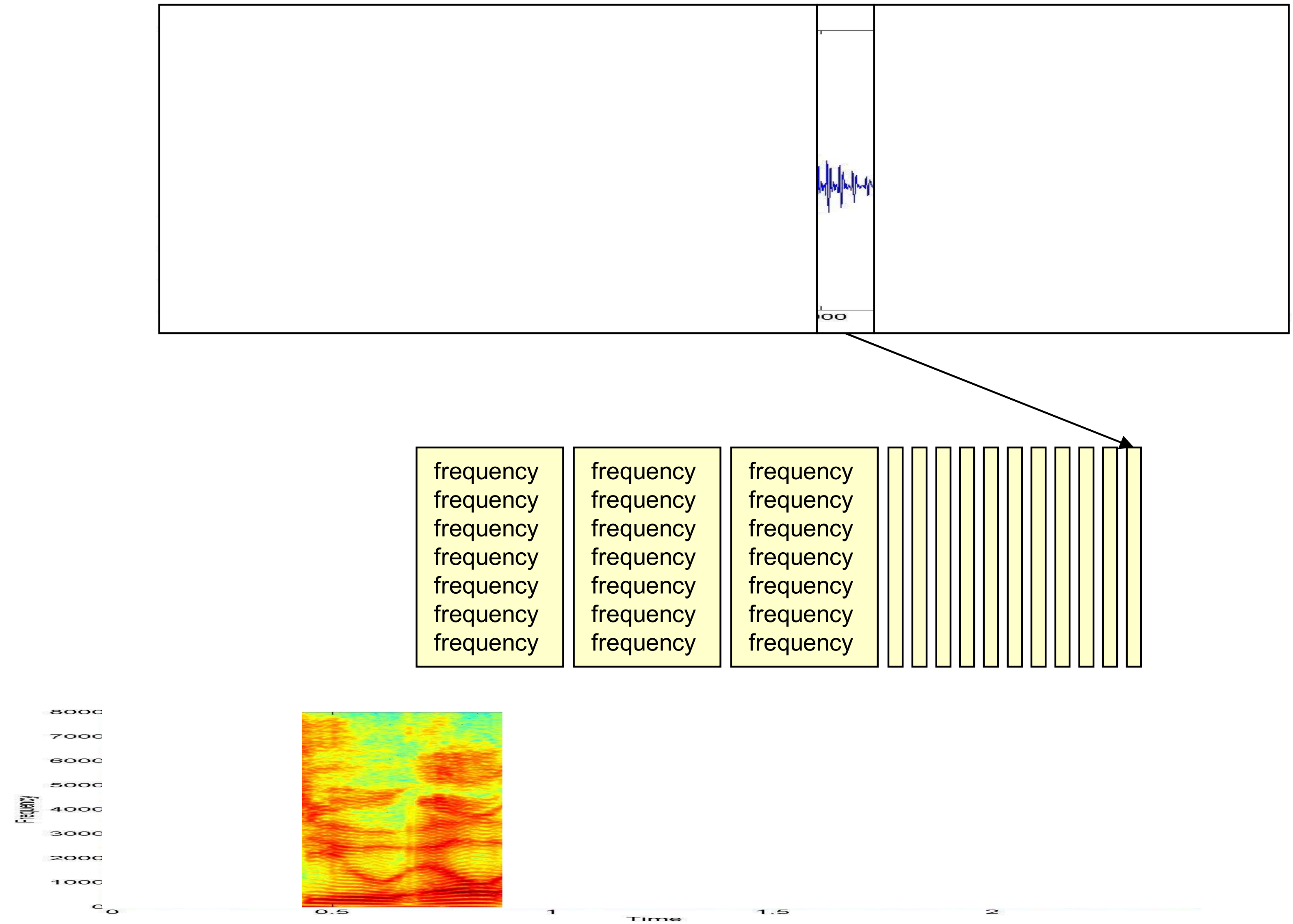
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



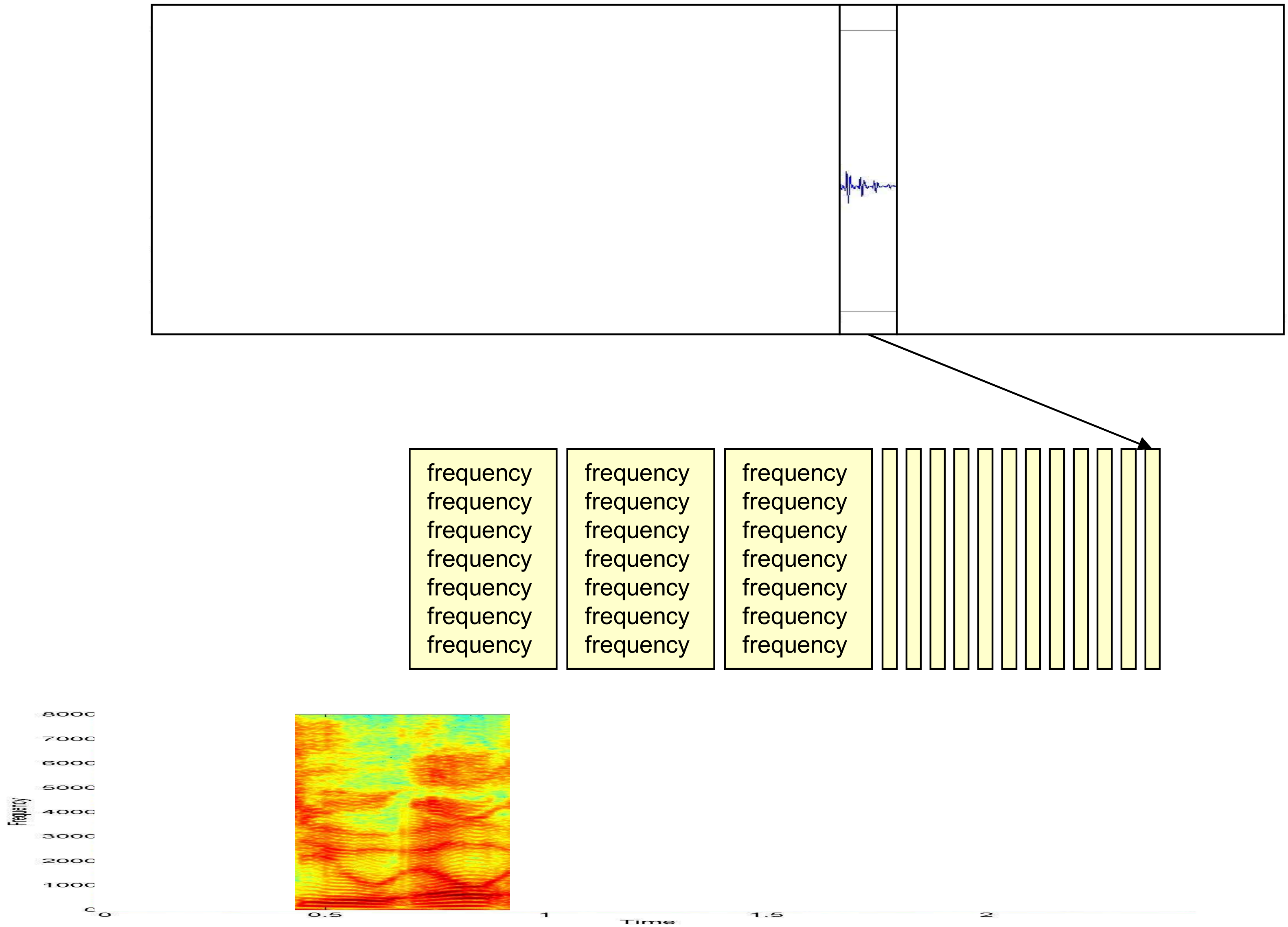
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



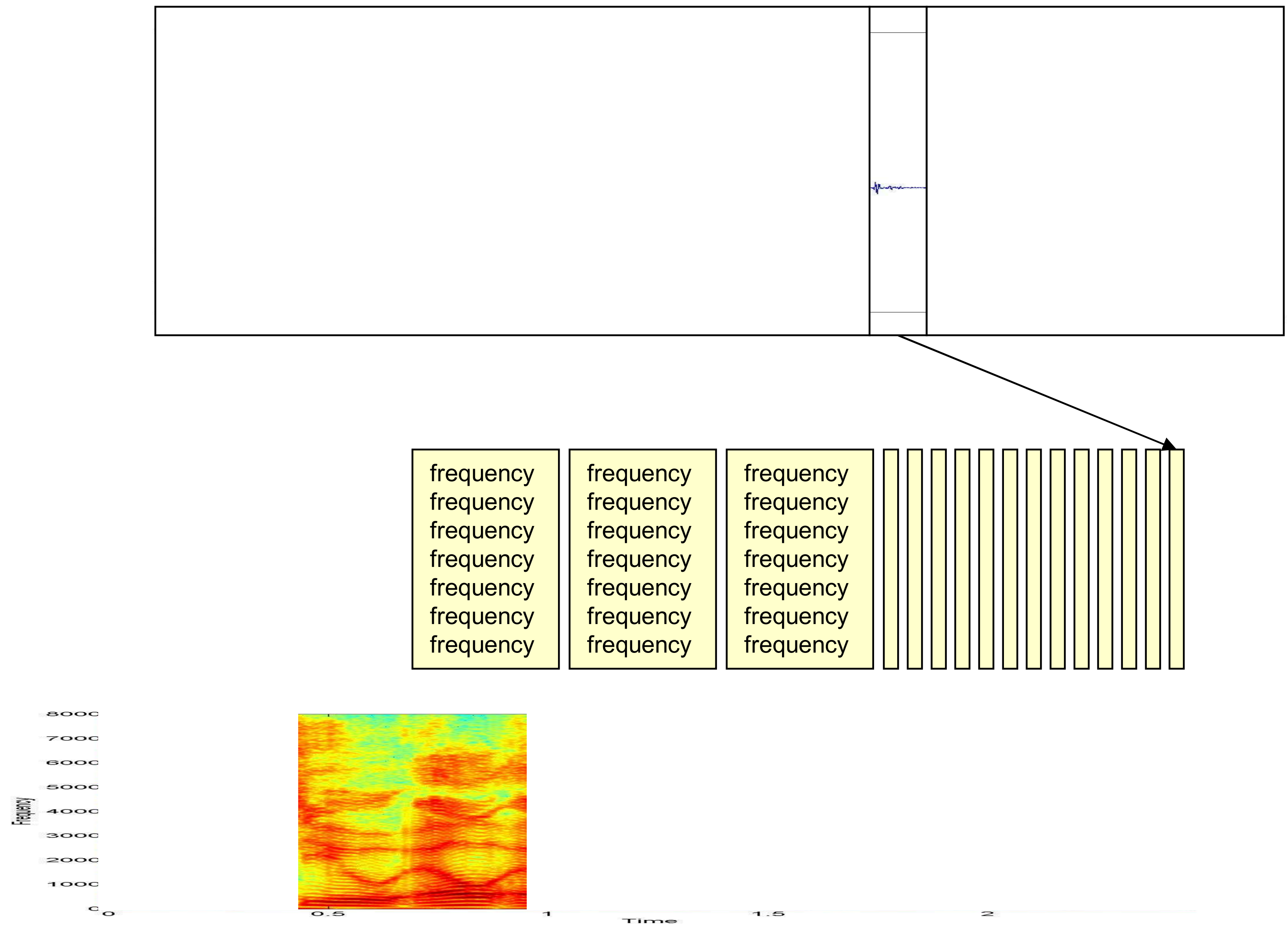
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



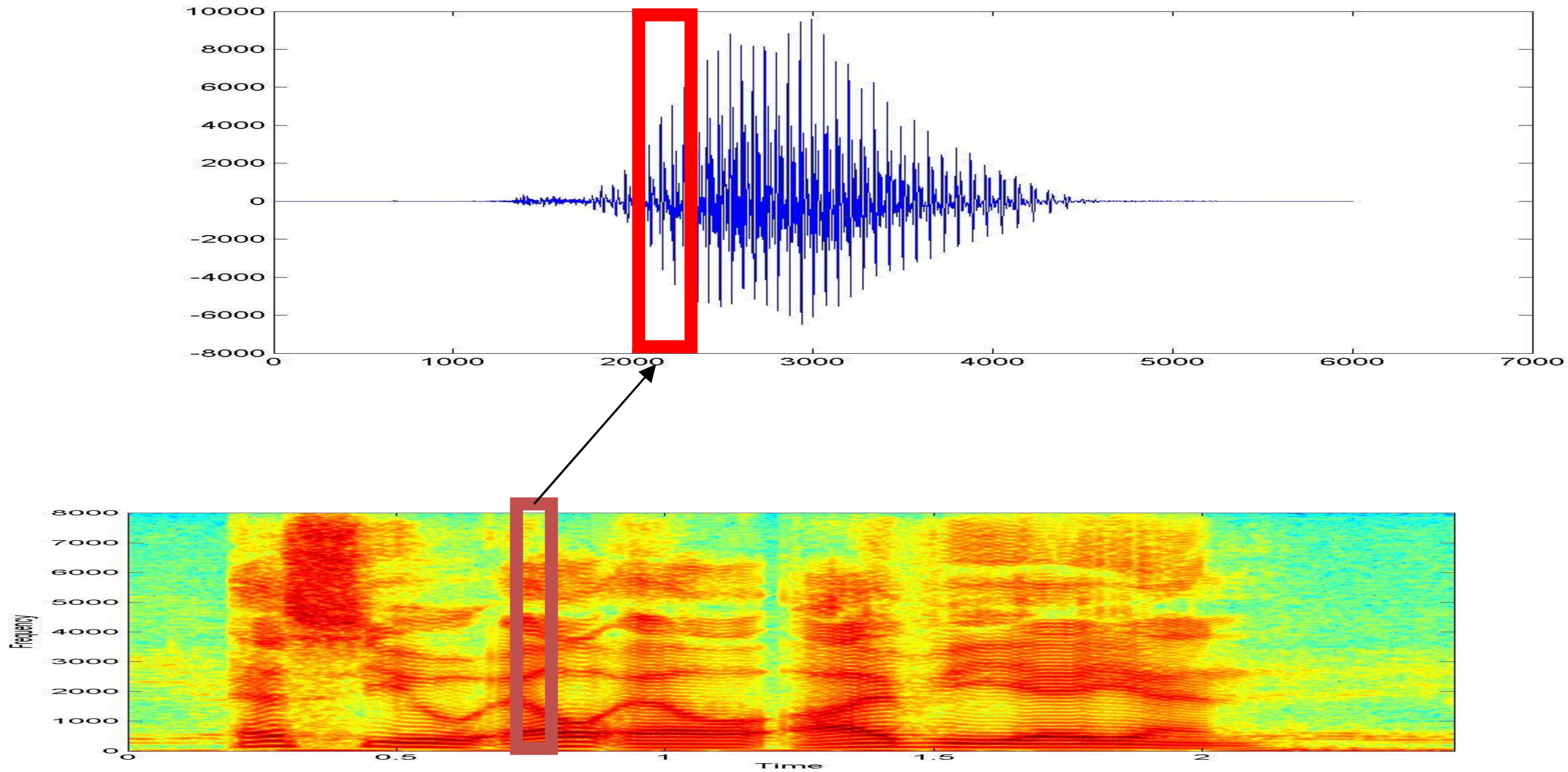
Compute Fourier Spectra of segments of audio and stack them side-by-side

Computing a Spectrogram



Compute Fourier Spectra of segments of audio and stack them side-by-side

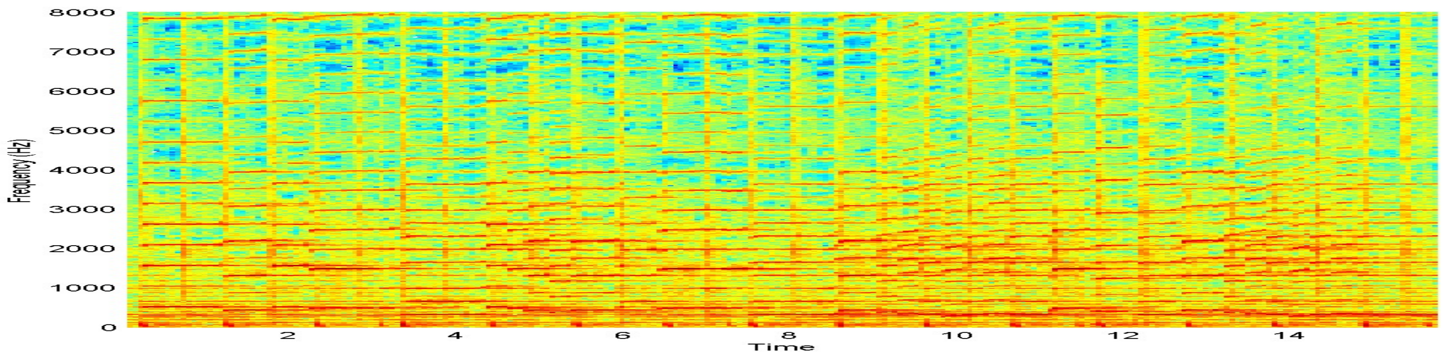
Computing the Spectrogram



Compute Fourier Spectra of segments of audio and stack them side-by-side
The Fourier spectrum of each window can be inverted to get back the signal.
Hence the spectrogram can be inverted to obtain a time-domain signal

In this example each segment was 25 ms long and adjacent segments overlapped by 15 ms

The result of parameterization



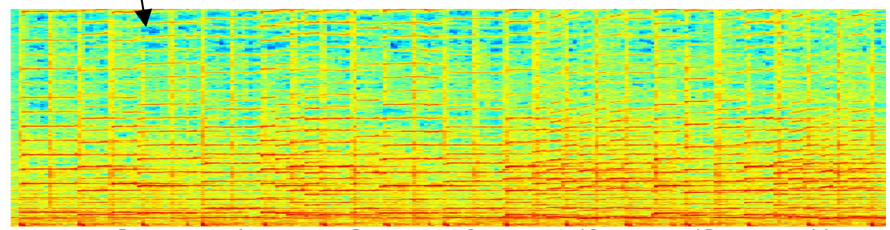
- Each column here represents the FT of a single segment of signal 64ms wide.
  - Adjacent segments overlap by 48 ms.
- DFT details
  - 1024 points (16000 samples a second).
  - 2048 point DFT – 1024 points of zero padding.
  - Only 1025 points of each DFT are shown
    - The rest are “reflections”
- The value shown is actually the magnitude of the complex spectral values
  - Most of our analysis / operations are performed on the magnitude

Magnitude and phase

?

$$S_k = |S_k| \exp(j.phase(S_k))$$

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



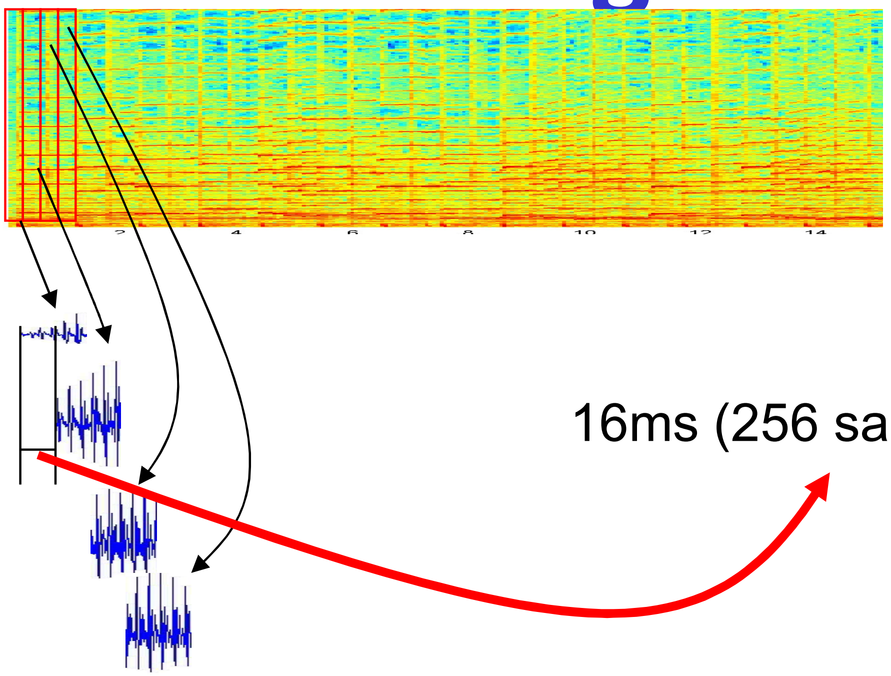
- All the operations (e.g. the examples shown in the previous class) are performed on the magnitude
- The phase of the complex spectrum is needed to invert a DFT to a signal
  - Where does that come from?
- Deriving phase is a serious, not-quite solved problem.

Phase

- Common tricks: Obtain the phase from the original signal
  - Sft = DFT(signal)
  - Phase1 = phase(Sft)
    - Each term is of the form real + j imag
    - For each element, compute arctan(imag/real)
  - Smagnitude = magnitude(Sft)
    - For each element compute Sqrt(real*real + imag*imag)
  - ProcessedSpectrum = Process(Smagnitude)
  - New SFT = ProcessedSpectrum*exp(j*Phase)
  - Recover signal from SFT

- Some other tricks:
  - Compute the FT of a different signal of the same length
  - Use the phase from that signal

Returning to the speech signal



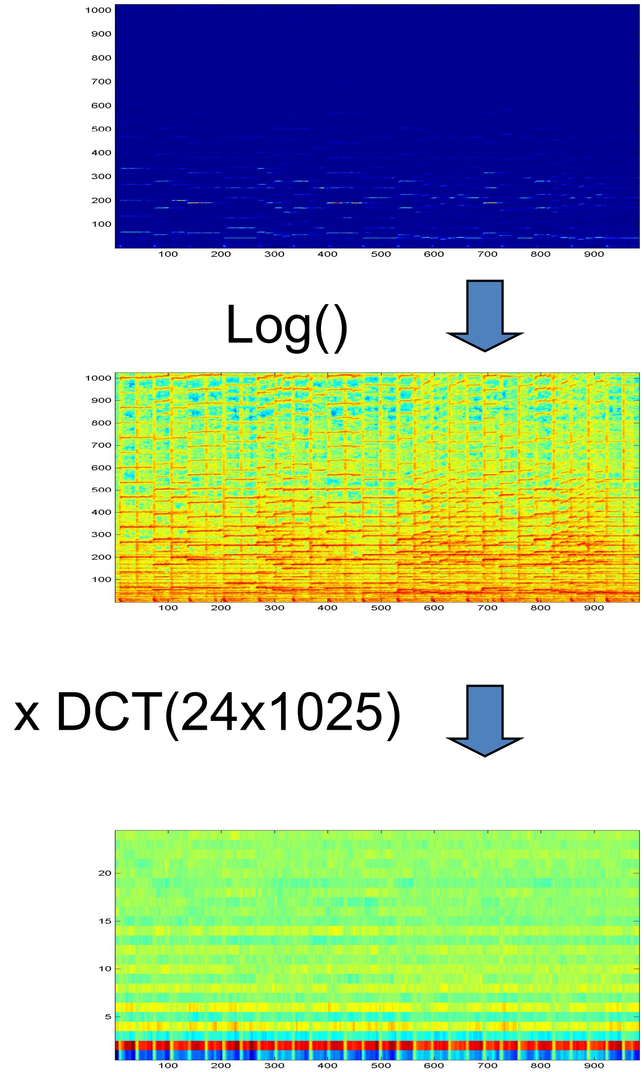
Actually a matrix of complex numbers

16ms (256 samples)

- For each complex spectral vector, compute a signal from the inverse DFT
  - Make sure to have the complete FT (including the reflected portion)
- If need be window the retrieved signal
- Overlap signals from adjacent vectors in exactly the same manner as during analysis
  - E.g. If a 48ms (768 sample) overlap was used during analysis, overlap adjacent segments by 768 samples

Additional tricks

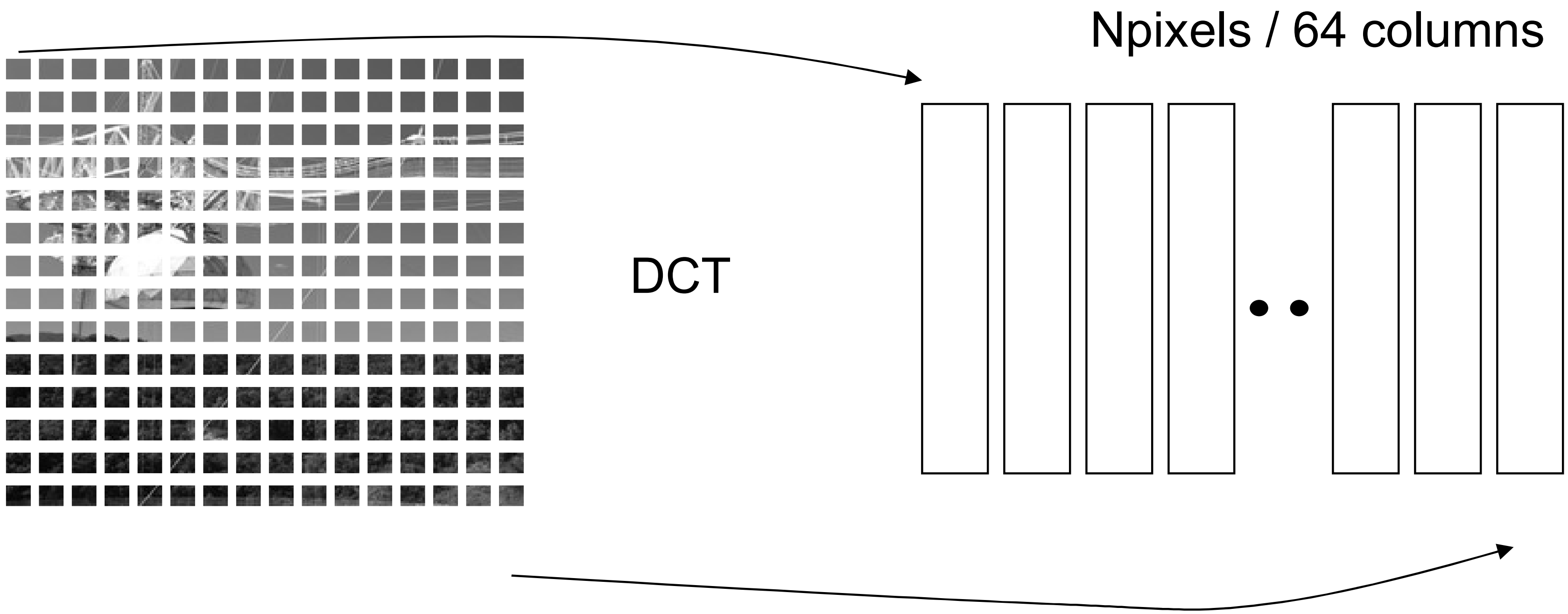
- The basic representation is the magnitude spectrogram
- Often it is transformed to a *log* spectrum
  - By computing the log of each entry in the spectrogram matrix
  - After processing, the entry is exponentiated to get back the magnitude spectrum
    - To which phase may be factored in to get a signal
- The log spectrum may be “compressed” by a dimensionality reducing matrix
  - Usually a DCT matrix



Index

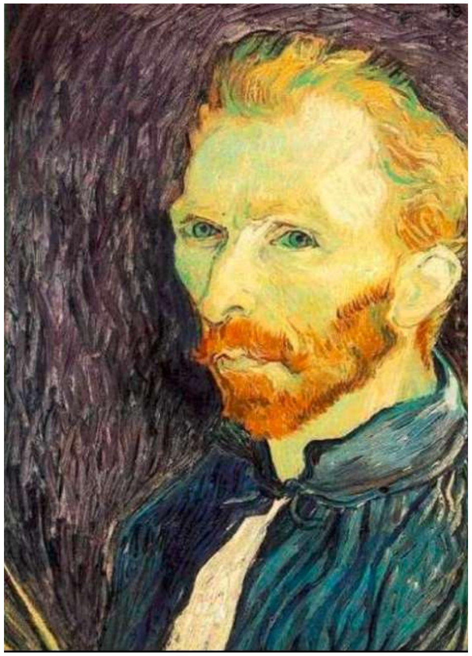
- Representing signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

Representing Images



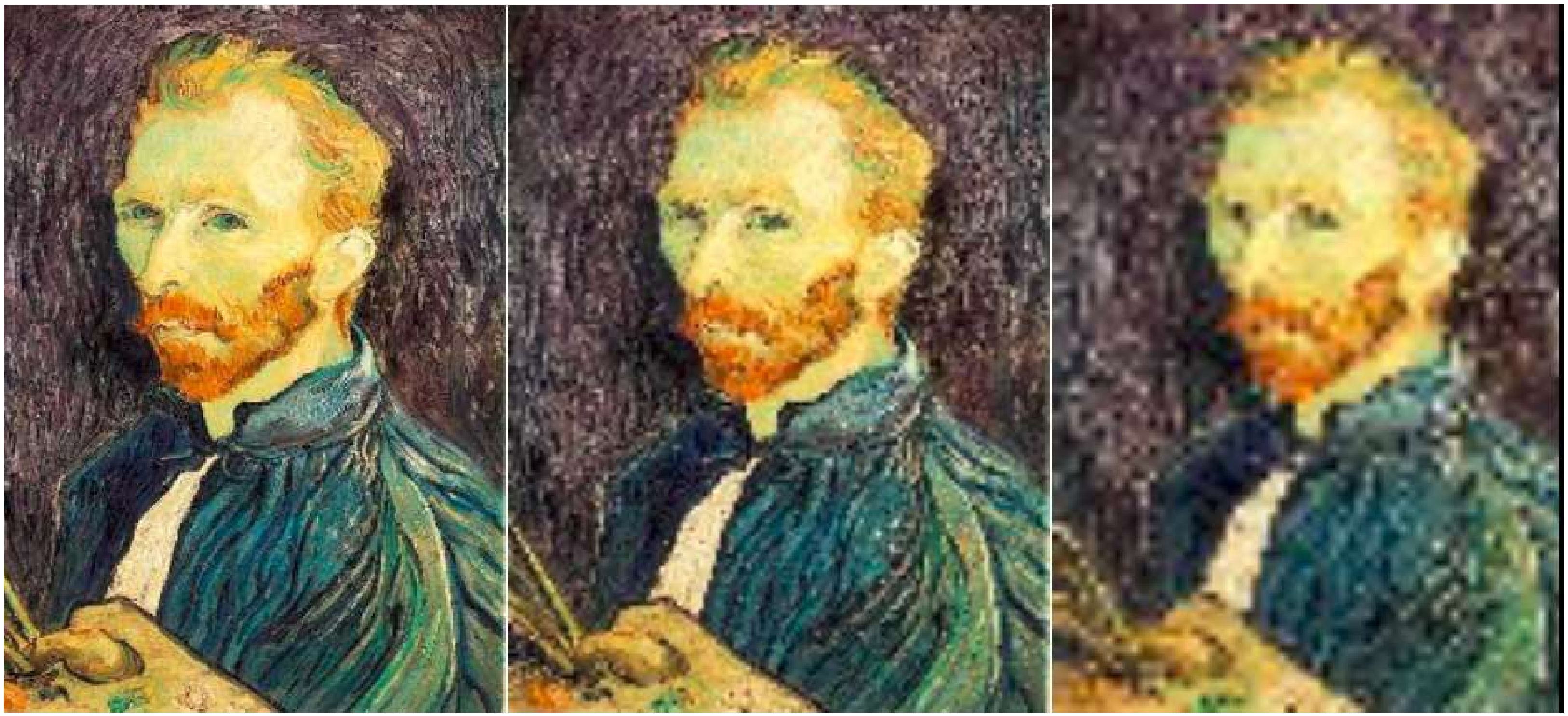
- DCT of small segments
  - 8x8
  - Each image becomes a matrix of DCT vectors
- DCT of the image

Downsampling-based representations



- Downsampling an example
  - Trying to reduce size by factor of 4 each time
    - Select every alternate sample row-wise and column-wisee
  - What exactly did we capture?
    - Clue : Results are horrible.

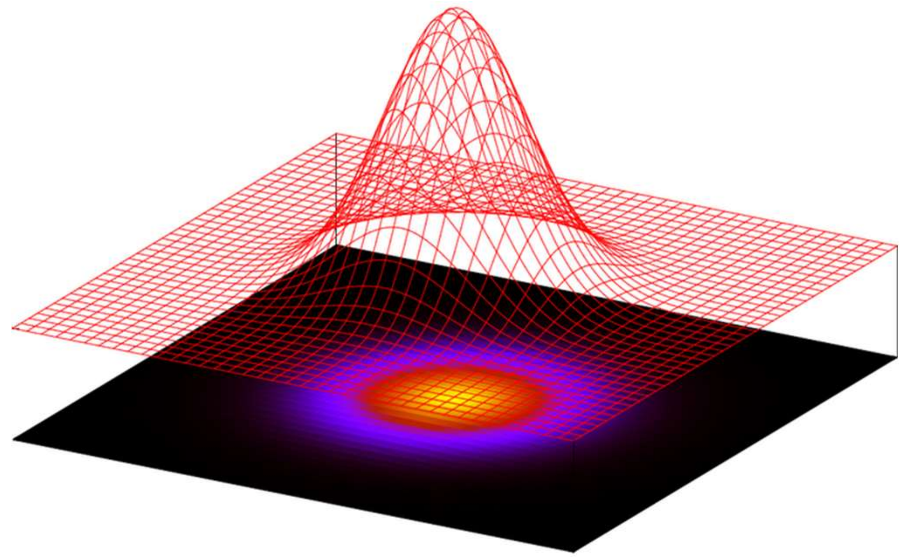
Downsampling-based representations



- Nasty aliasing effects!

The Gaussian Kernel

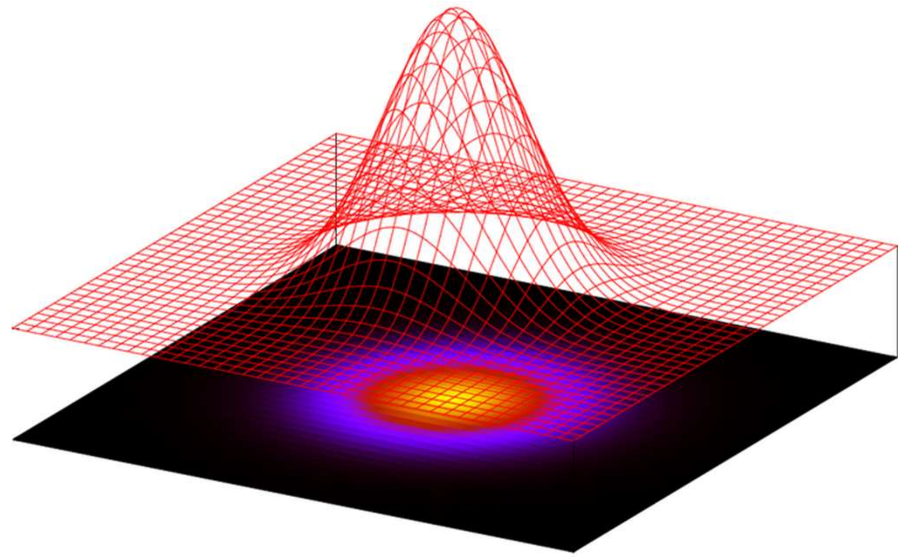
$$\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_N \end{bmatrix}$$



- A two-dimensional image of a Gaussian
- Characterized by
  - Center (mean)
  - Standard deviation σ (assumed same in both directions)
    - I.e. sphereical Gaussian
- The image can be represented by a vector

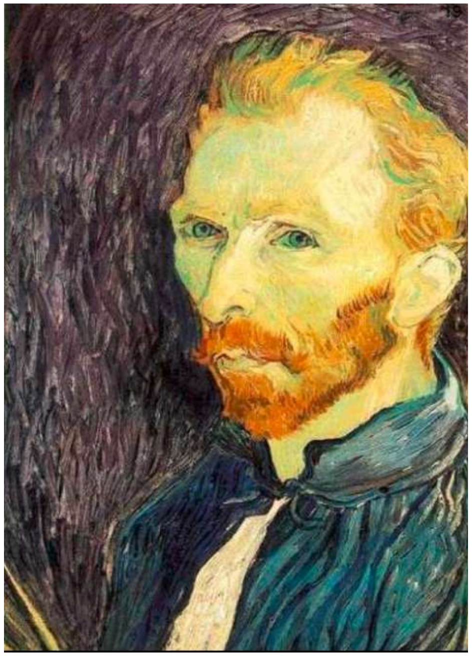
The Gaussian Kernel matrix

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N1} & g_{N2} & \cdots & g_{NN} \end{bmatrix}$$



- Each column is one Gaussian
  – Representing a Gaussian centered at one of the pixels in the image
- As many columns as pixels
  – Also as many rows as pixels

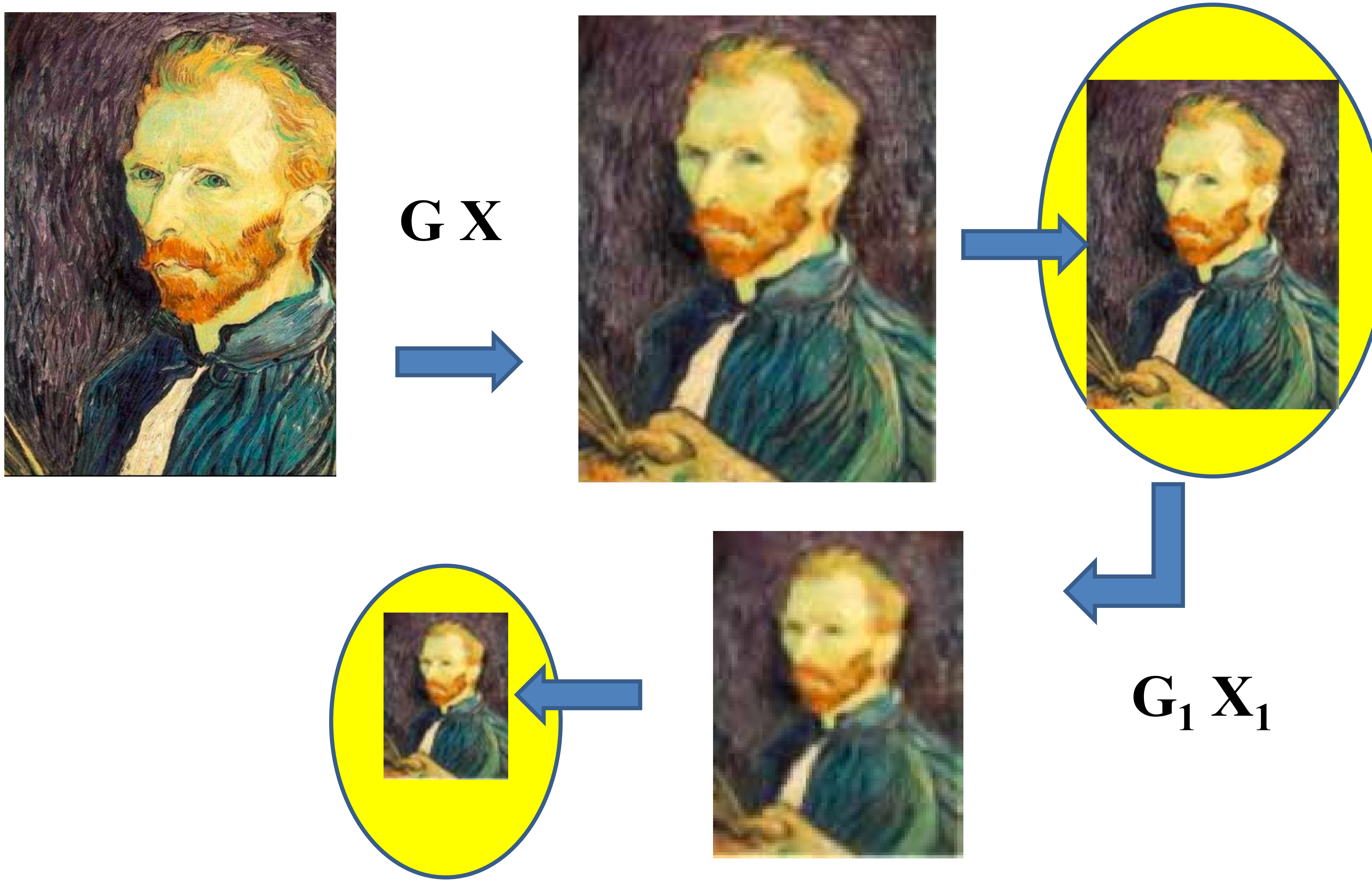
Downsampling-based representations



$$\mathbf{G}\,\mathbf{X} \qquad X = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}$$

- Transform with Gaussian kernel matrix
- Then downsample

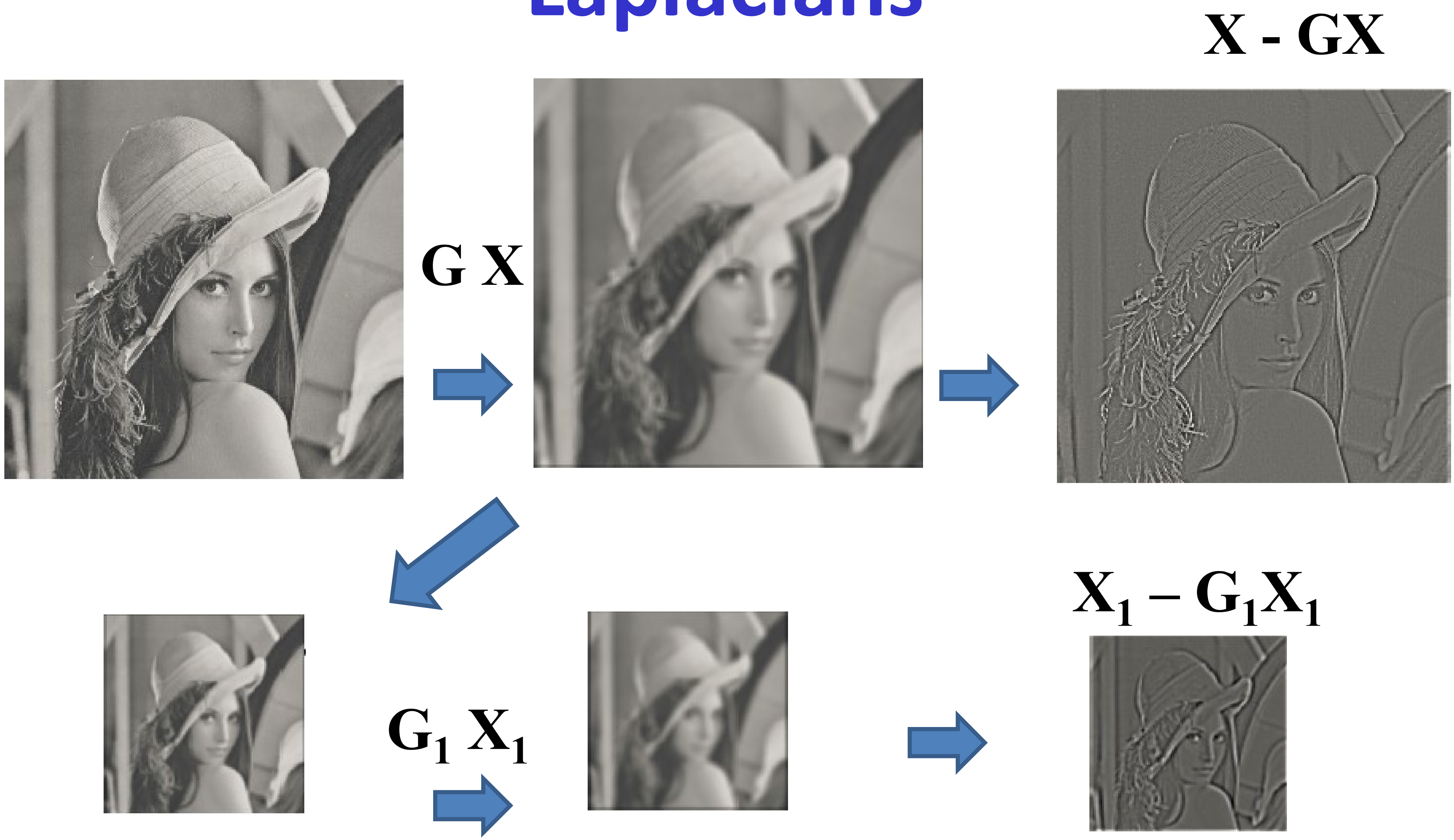
Downsampling-based representations



The Gaussian Pyramid

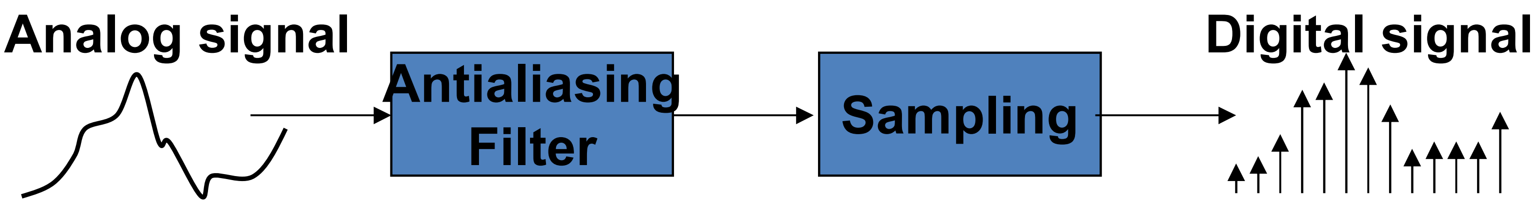
- Successive smoothing and scaling
- The entire collection of images is the Gaussian pyramid

Laplacians



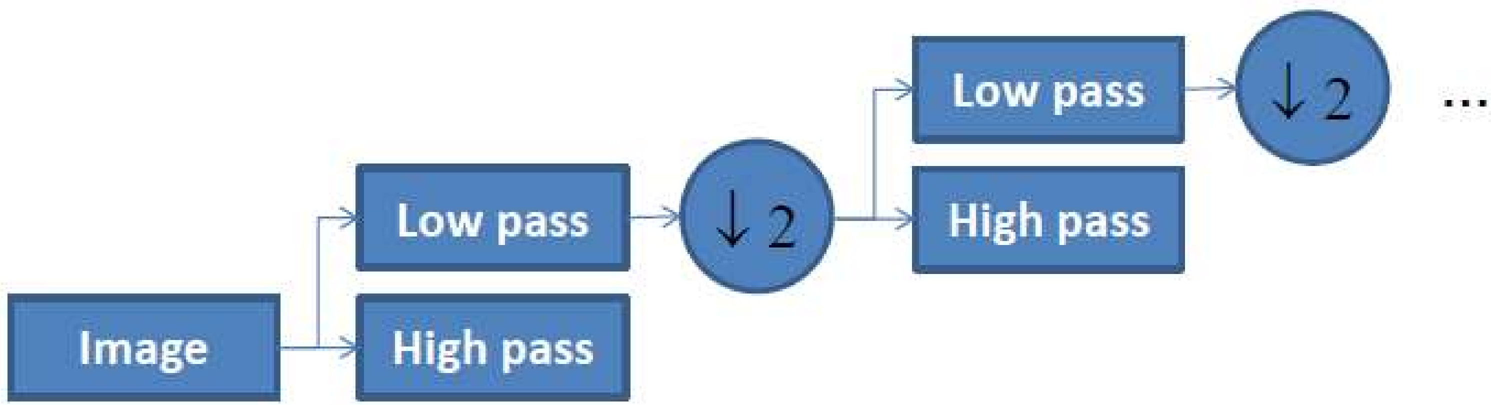
Laplacian Pyramid

Remember..



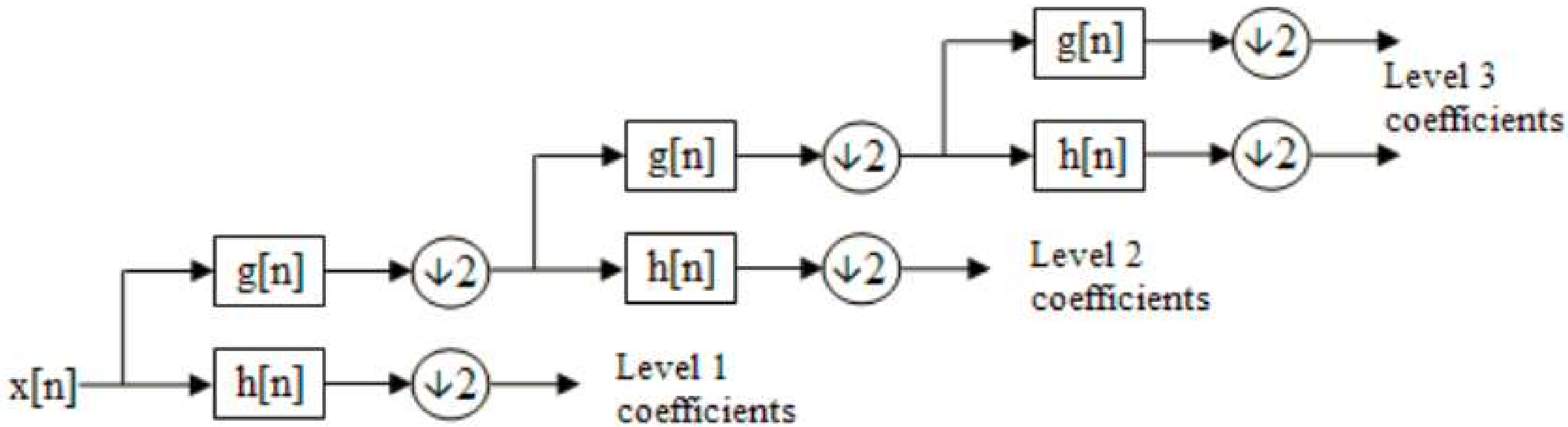
- The Gaussian is an anti-aliasing filter
- The Gaussian pyramid is the *low-pass filtered* version of the image
- The Laplacian pyramid is the *high-pass filtered* version of the image

The Gaussian/Laplacian Decomposition



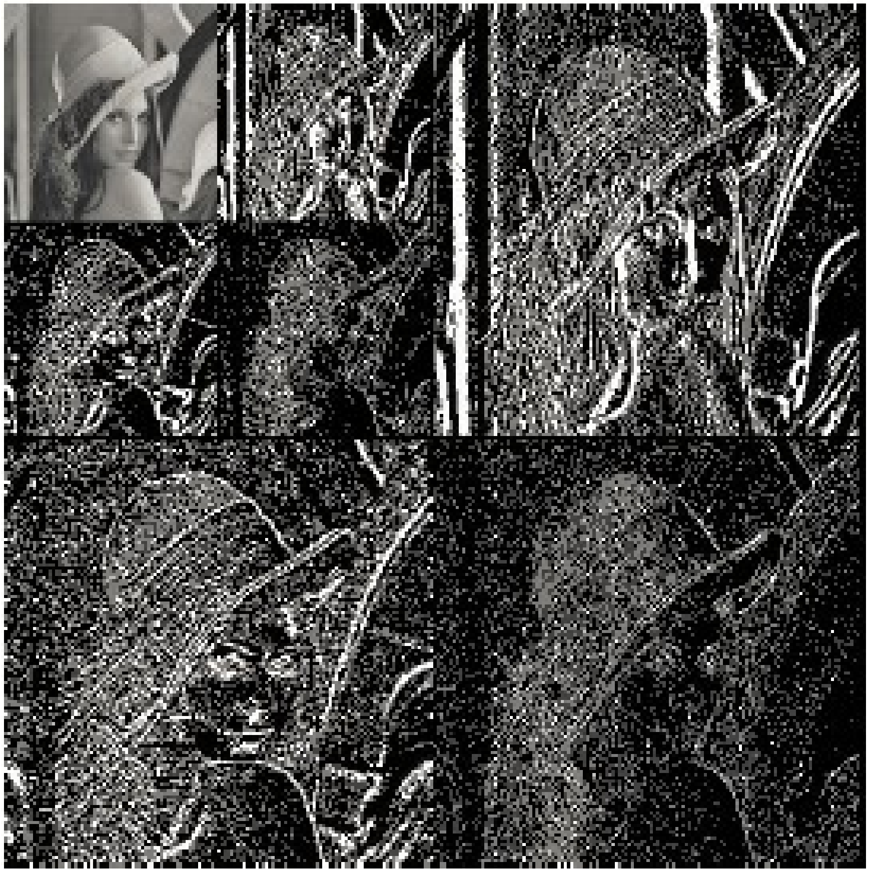
- Each low-pass filtered image is downsampled
- The process is recursively performed

The discrete wavelet transform



- Very similar in structure
- But the bases at each scale are orthogonal to bases at other scales
  - As opposed to a Gaussian kernel matrix

Haar Wavelets



- We have already encountered Haar wavelets

Other characterizations

- Content-based characterizations
  - E.g. Hough transform
    - Captures linear arrangements of pixels
  - Radon transform
  - SIFT features
  - Etc.

Summary

- The need to represent signals
- Basis-based representations
- Haar bases
  - For images and sound
- Fourier bases
  - For images and sound
    - Generalizes to any time-series signal or 2D signal
- Spectrograms
  - For sound and time-series data
- Real Fourier representations, aka DCT
  - For sound and images
- Gaussian and Laplacian pyramids for images

Next up..

- The representations we saw today were *deterministic*
- The bases were designed without considering the specific data set
- Next: data-dependent bases