Fourier Transforms – Imaging Applications: Compression and Aliasing

CS370 Lecture 24 – March 10, 2017

1D Grayscale Image

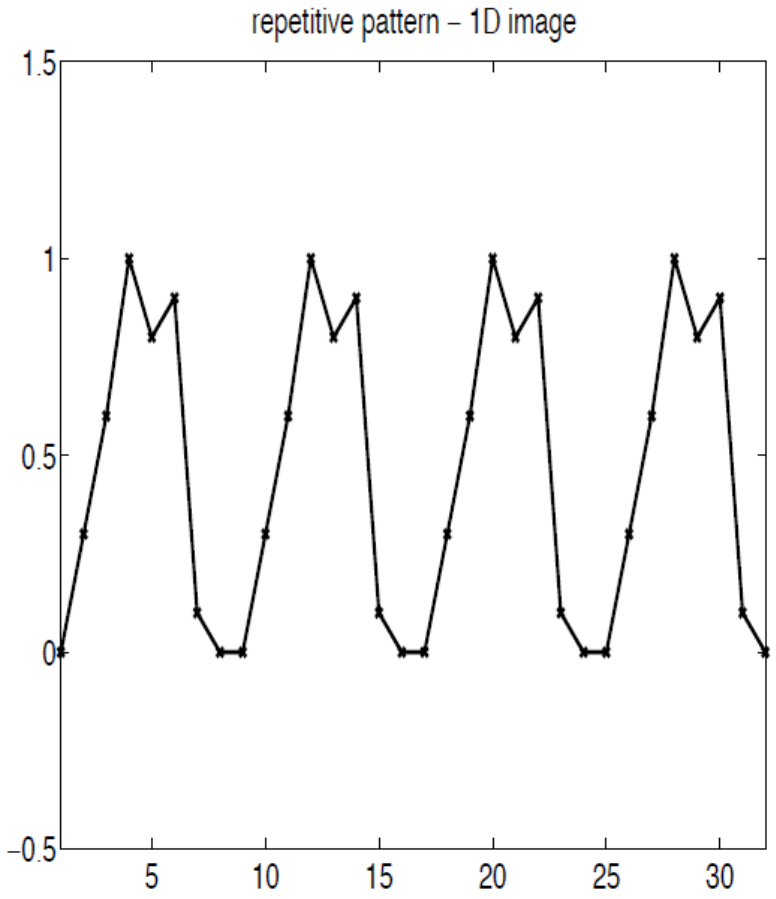
We can think of a 1D array of grayscale values as a 1D image.

Each entry gives the pixels' intensity/gray values

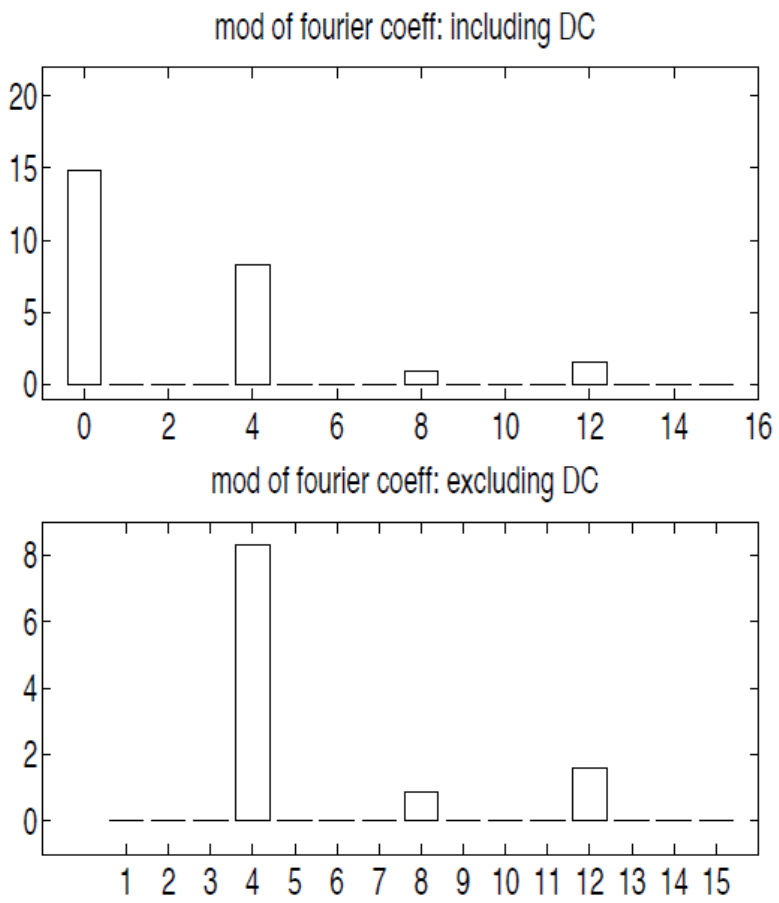
(Or you might imagine this as a single row of a 2D grayscale image.)

Processing “1D Images”

For grayscale images, we can perform a DFT on the pixels’ intensity/gray values.



1-D image x



Size of Fourier Coefficients

Processing “1D Images”

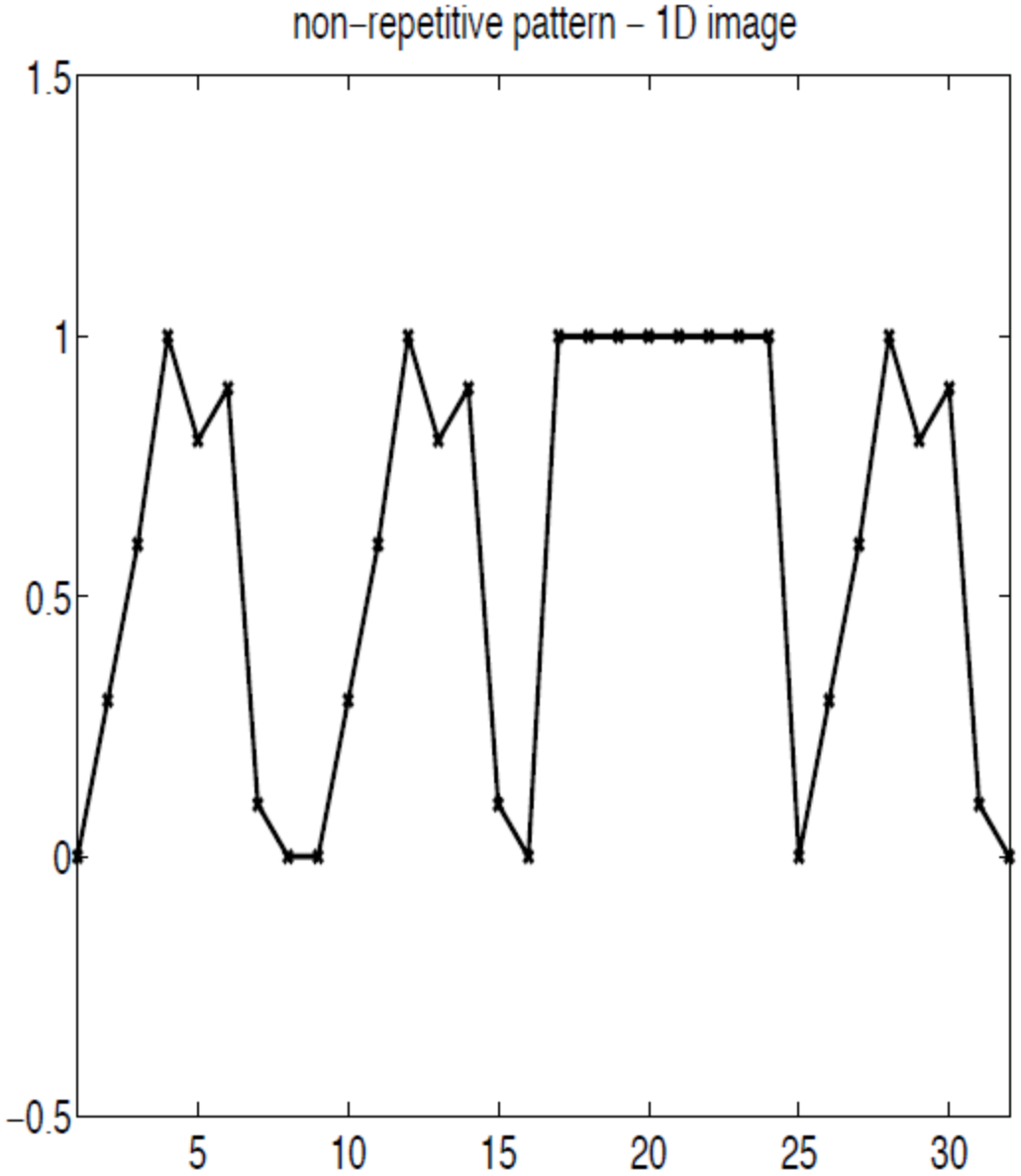
In this example…

- Average is about 0.5, so $F_0 \approx 0.5$. (Well, actually ~15 since plot uses Matlab’s convention.)
- The dominant pattern repeats four times, so coefficient F_4 is large.
- Overall pattern is very repetitive, so *few* coefficients are active.

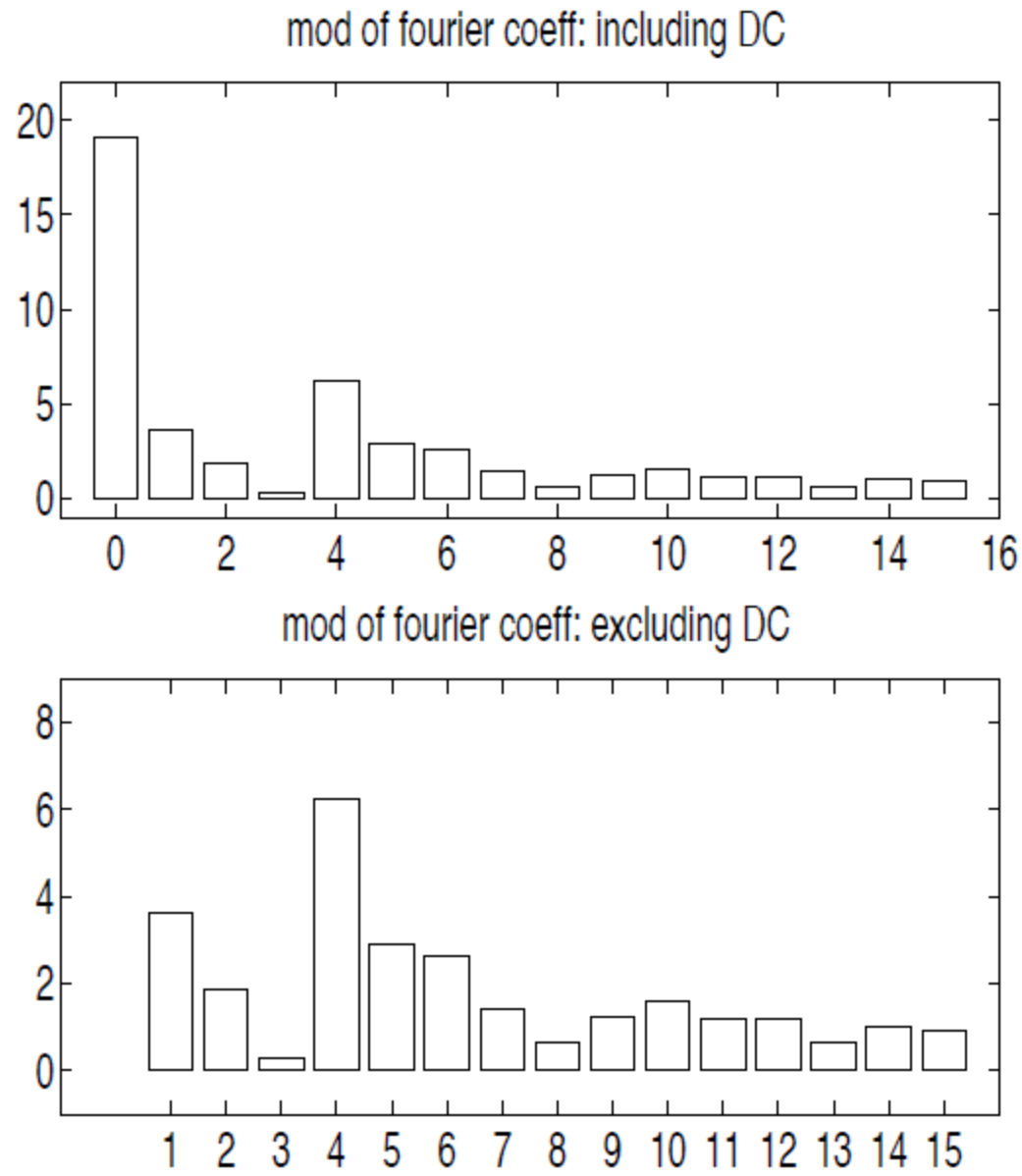
We can store it cheaply just with F_0, F_4, F_8 and F_{12}.

If we instead replace the 3rd bump’s top with a flat line, the simple repetition is destroyed.

- Many more Fourier coefficients will become active / non-zero. Expensive to store!



1-D image y



Size of Fourier Coefficients

Compression of 1D Images

- Although many coefficients are non-zero, many still have fairly small moduli/magnitude…
- …so those frequencies contribute less to the image.

Compression Strategy:

- Create an (approximate) compressed version of the image, f_n, by throwing away "small" Fourier coefficients: $|F_k| < tol$.
- To reconstruct the image, run the **inverse** DFT to get modified data (pixels), $\widehat{f_n}$.
- Discard the imaginary parts of $\widehat{f_n}$, to ensure new data is strictly real.

Comparison:

Recovered “image” hopefully has fairly small deviations from the original “true” data.

But! We store fewer Fourier coefficients and so save space!

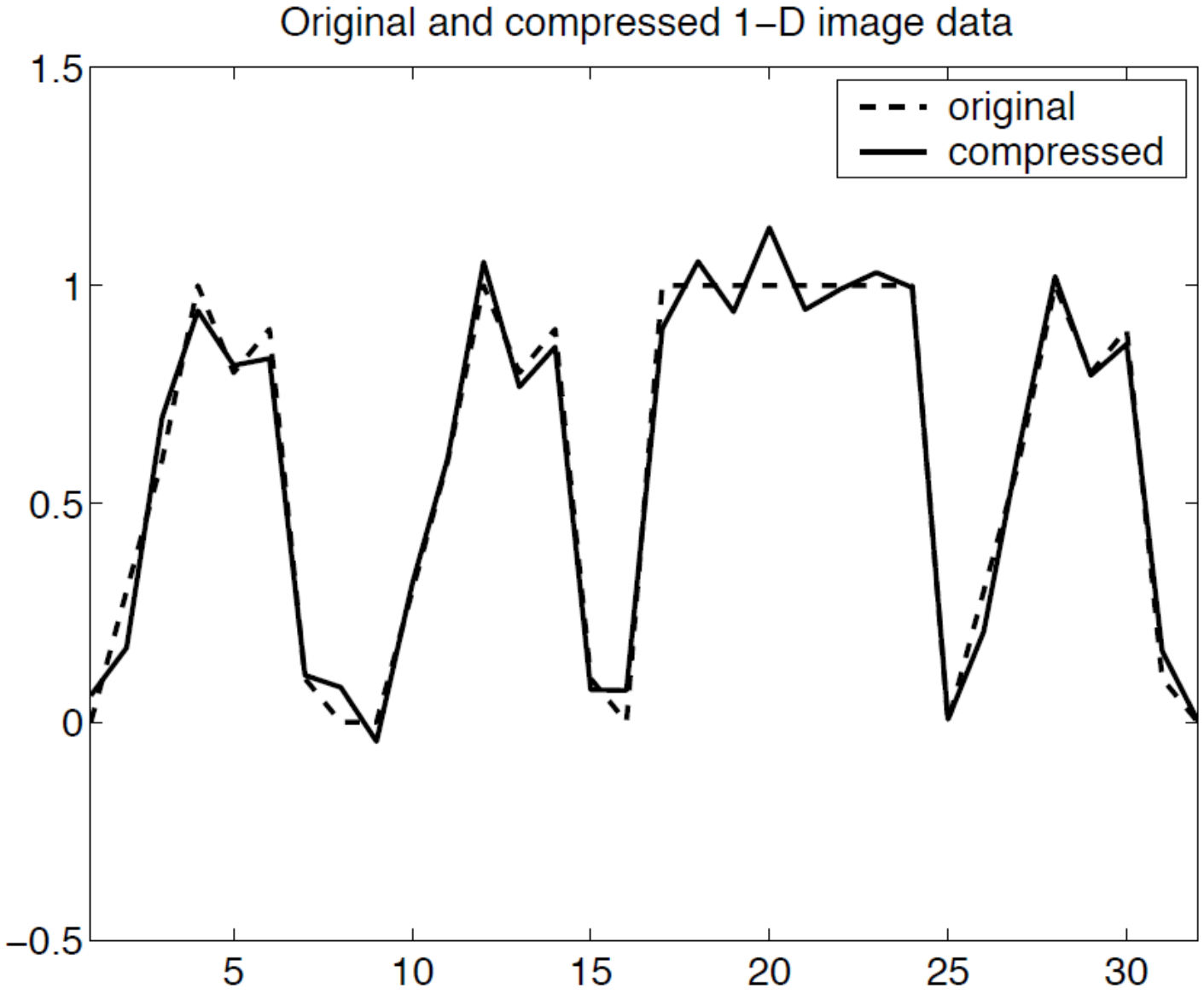


Image Compression in 2D

Ultimately, this is what we would like to achieve…

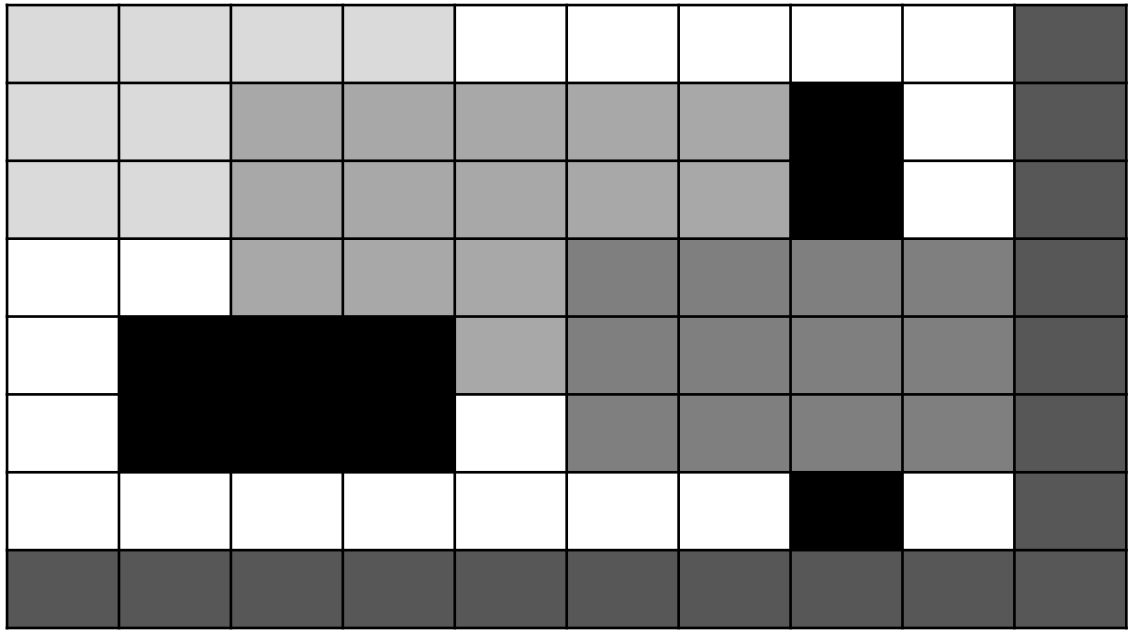
(a) Original

(b) Compressed by 85%

Image Processing in 2D

How does the DFT work for regular 2D (grayscale) image data?

i.e., we have a 2D array X of per-pixel intensities, of size $M \times N$.
Assume we have *scaled* image data, so $0 \leq X(i,j) \leq 1$.



2D Discrete Fourier Transform

We apply a 2D extension of the DFT to the data, like so:

$$F_{k,l} = \frac{1}{NM} \sum_{n=0}^{N-1} \sum_{j=0}^{M-1} f_{n,j} W_N^{-nk} W_M^{-jl}$$

Essentially two standard (1D) DFT's combined.

Two (distinct) roots of unity are used, one per dimension:

$$W_N = e^{\frac{i2\pi}{N}} \text{ and } W_M = e^{\frac{i2\pi}{M}}$$

Result is a 2D array of Fourier coefficients, $F_{k,l}$.

2D *Fast* Fourier Transform

The 2D ***FFT*** can be computed *efficiently* using nested 1D FFTs:

- First, convert each row using 1D FFTs.
- Then, convert each column of the result, again using 1D FFTs.

Matlab has **fft2** and **ifft2**

(As usual, watch for scaling convention differences.)

2D Fast Fourier Transform

Visually:

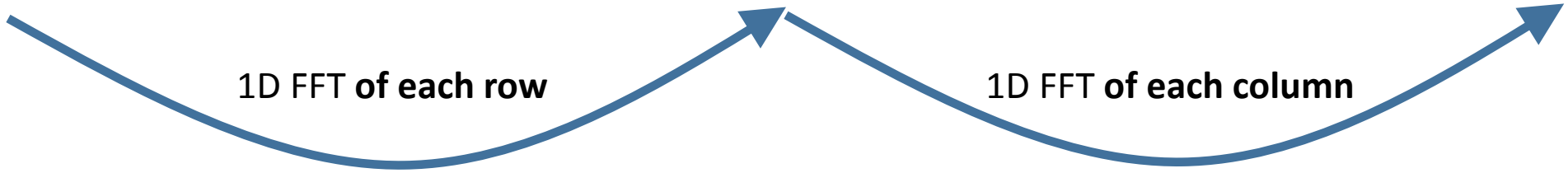
Input Data

$f_{0,0}$	$f_{0,1}$	$f_{0,2}$	…	$f_{0,M-1}$
$f_{1,0}$	$f_{1,1}$	$f_{1,2}$	…	$f_{1,M-1}$
$f_{2,0}$	$f_{2,1}$	$f_{2,2}$	…	$f_{2,M-1}$
…	…	…	…	…
$f_{N-1,0}$	$f_{N-1,1}$	$f_{N-1,2}$	…	$f_{N-1,M-1}$

$H_{0,0}$	$H_{0,1}$	$H_{0,2}$	…	$H_{0,M-1}$
$H_{1,0}$	$H_{1,1}$	$H_{1,2}$	…	$H_{1,M-1}$
$H_{2,0}$	$H_{2,1}$	$H_{2,2}$	…	$H_{2,M-1}$
…	…	…	…	…
$H_{N-1,0}$	$H_{N-1,1}$	$H_{N-1,2}$	…	$H_{N-1,M-1}$

Fourier Coefficients

$F_{0,0}$	$F_{0,1}$	$F_{0,2}$	…	$F_{0,M-1}$
$F_{1,0}$	$F_{1,1}$	$F_{1,2}$	…	$F_{1,M-1}$
$F_{2,0}$	$F_{2,1}$	$F_{2,2}$	…	$F_{2,M-1}$
…	…	…	…	…
$F_{N-1,0}$	$F_{N-1,1}$	$F_{N-1,2}$	…	$F_{N-1,M-1}$



2D FFT via 1D FFTs - Derivation

$$F_{k,l} = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{j=0}^{M-1} f_{n,j} W_N^{-nk} W_M^{-jl}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} W_N^{-nk}\left(\frac{1}{M}\sum_{j=0}^{M-1} f_{n,j} W_M^{-jl}\right)$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} H_{n,l} W_N^{-nk} \qquad \text{where we defined } H_{n,l} = \frac{1}{M}\sum_{j=0}^{M-1} f_{n,j} W_M^{-jl}$$

This is a 1D FFT (per row) to get H.

This is a 1D FFT (per column) on H to get F.

2D Fast Fourier Transform - Complexity

Performing M 1D FFT's of length N: $M \cdot O(Nlog_2N) = O(MNlog_2N)$.

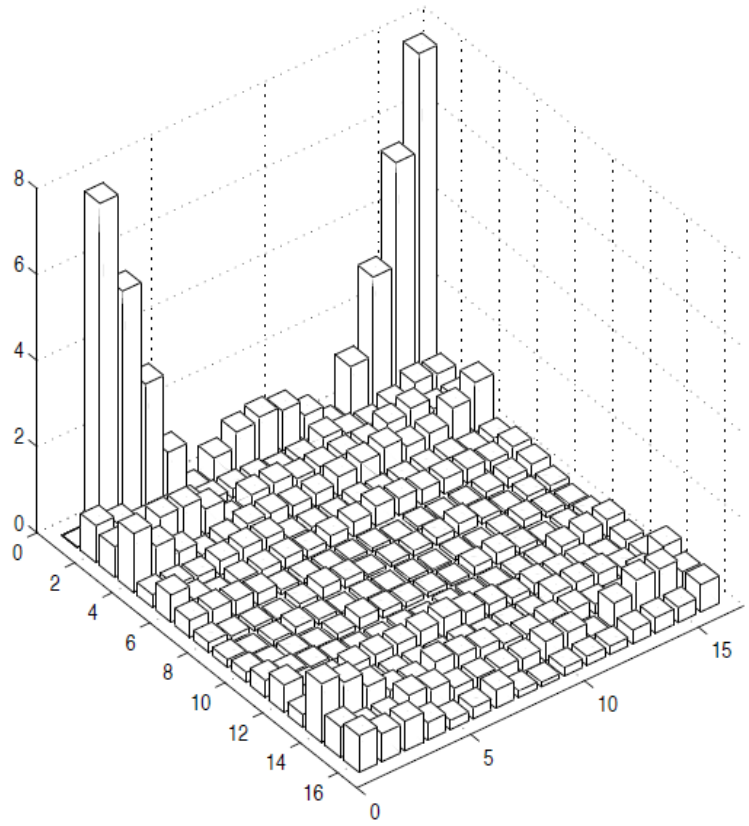
Performing N 1D FFT's of length M: $N \cdot O(Mlog_2M) = O(MNlog_2M)$.

So total complexity is $O(MN(\log_2 M + \log_2 N))$.

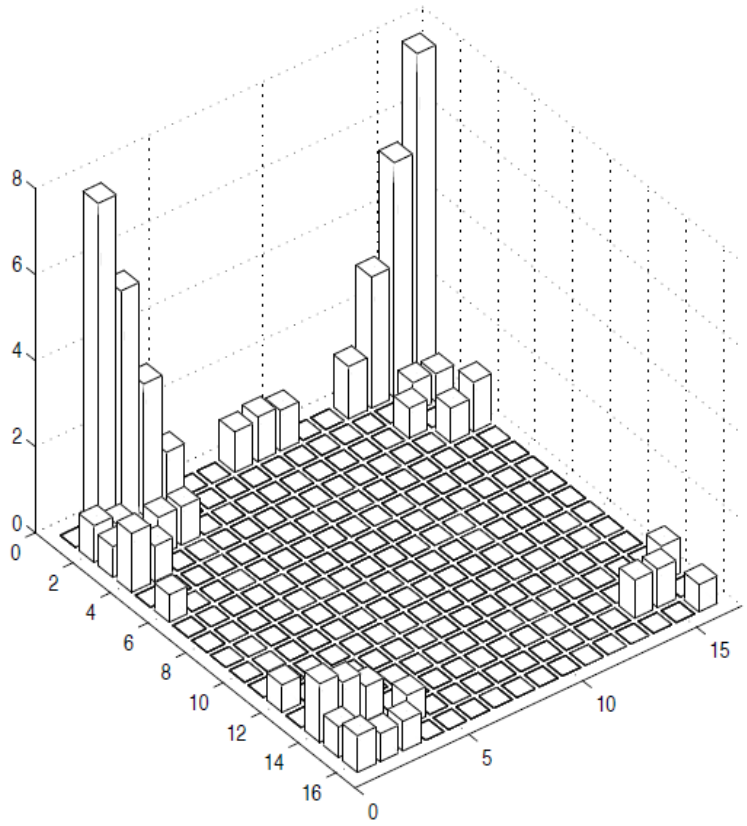
Simple Image Compression in 2D:

Given the 2D FFT of image intensities, threshold the moduli of the Fourier coefficients to discard small ones, just like in 1D.

2D Fourier Plot for a 16x16 block



(a) Original

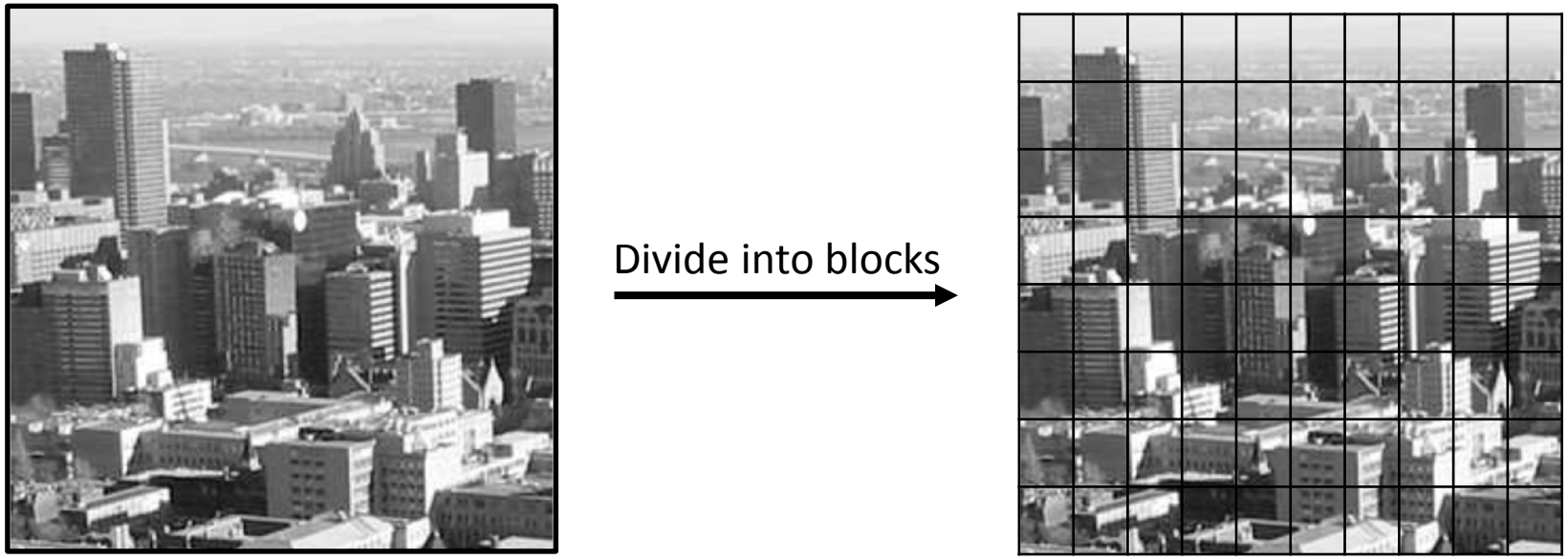


(b) Compressed by 85%

Block-based Image Compression

Often an image is subdivided into smaller blocks: 16x16 or 8x8 pixels.

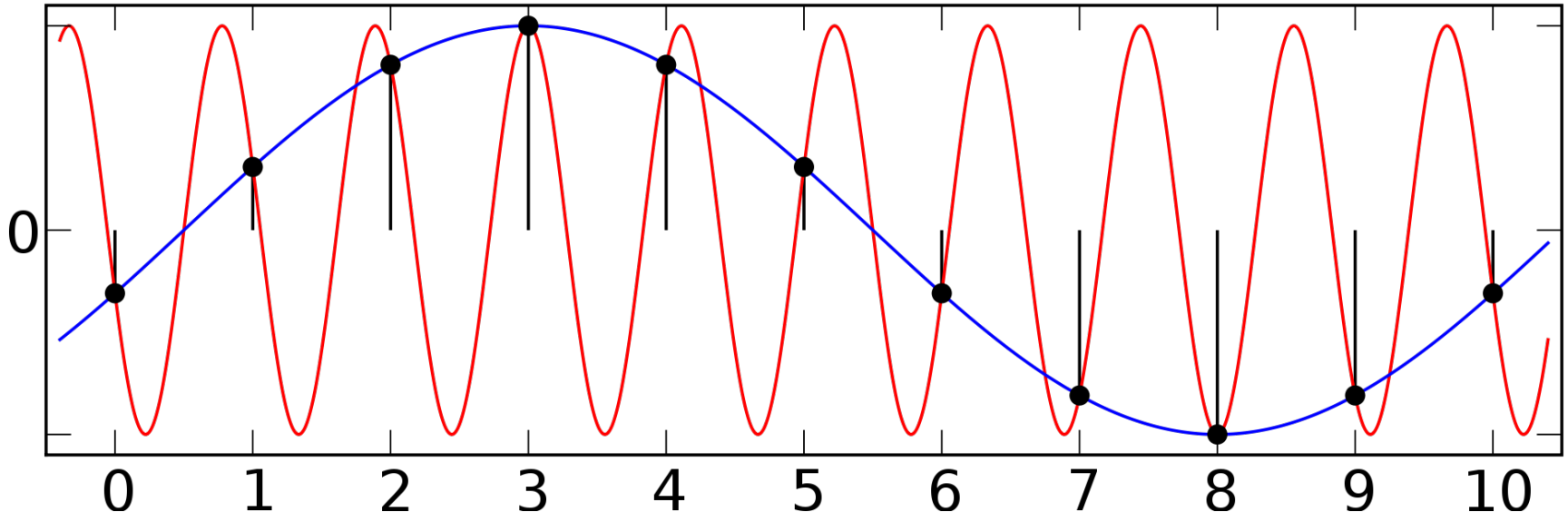
Compression is performed on each block independently.



Pixels within a block often have similar/related data, and hopefully compress more effectively.

FFT Issue: Aliasing

If a input signal contains high frequencies, but the spacing of our sampled data points is inadequate, "***aliasing***" can occur.



A truly high frequency signal (red) ***aliases as*** (appears to be) a lower frequency signal (blue) in the discrete result!

Aliasing in Action: (Still) Photography

Desired Image

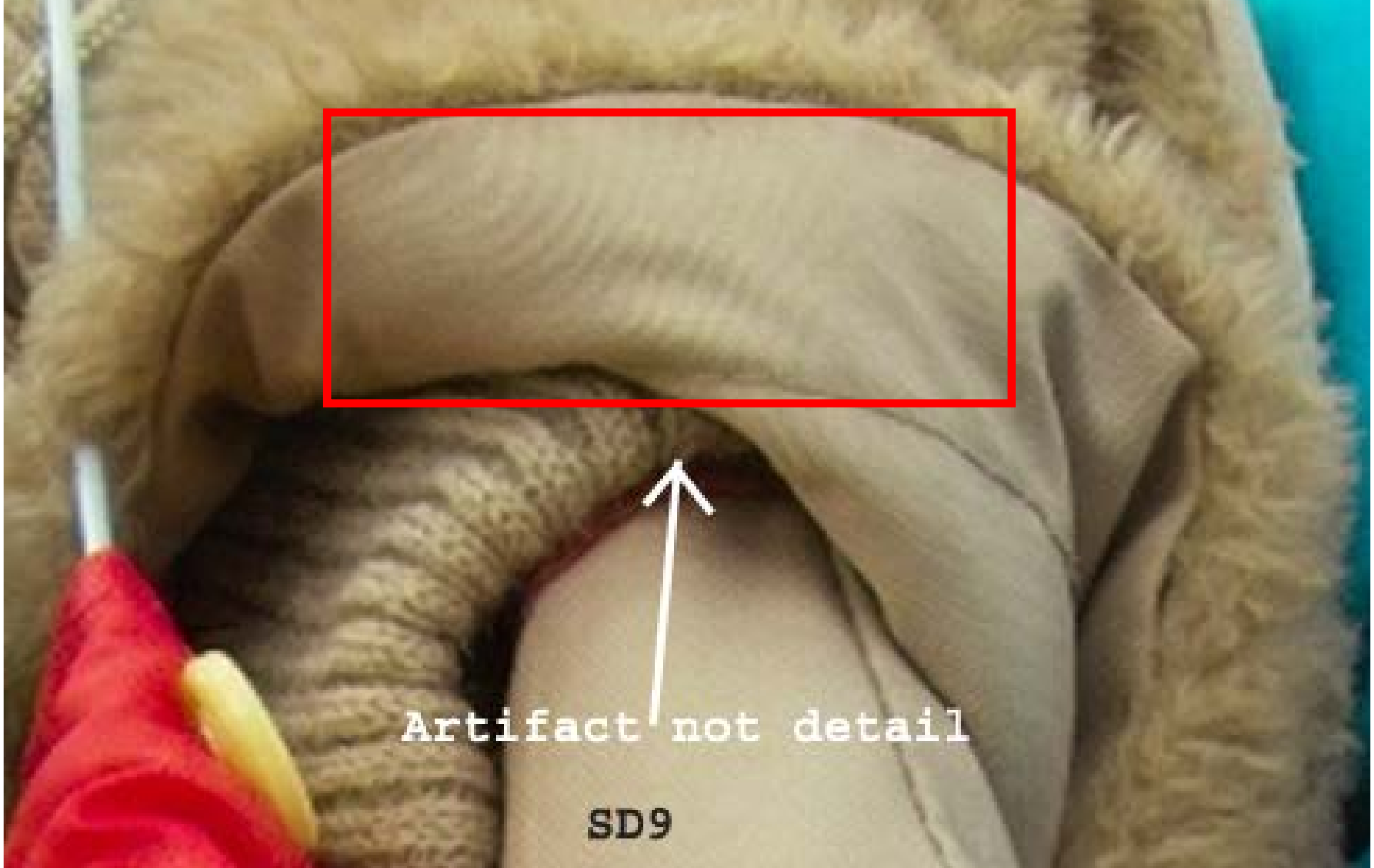
Captured Image, with Aliasing

(Images by Dave Etchells.)

Sampling signals without treating aliasing can lead to errors in the captured discrete data.

Right image: artificial visual patterns (“Moire patterns”) appear that are not present in the real material, due to aliasing.

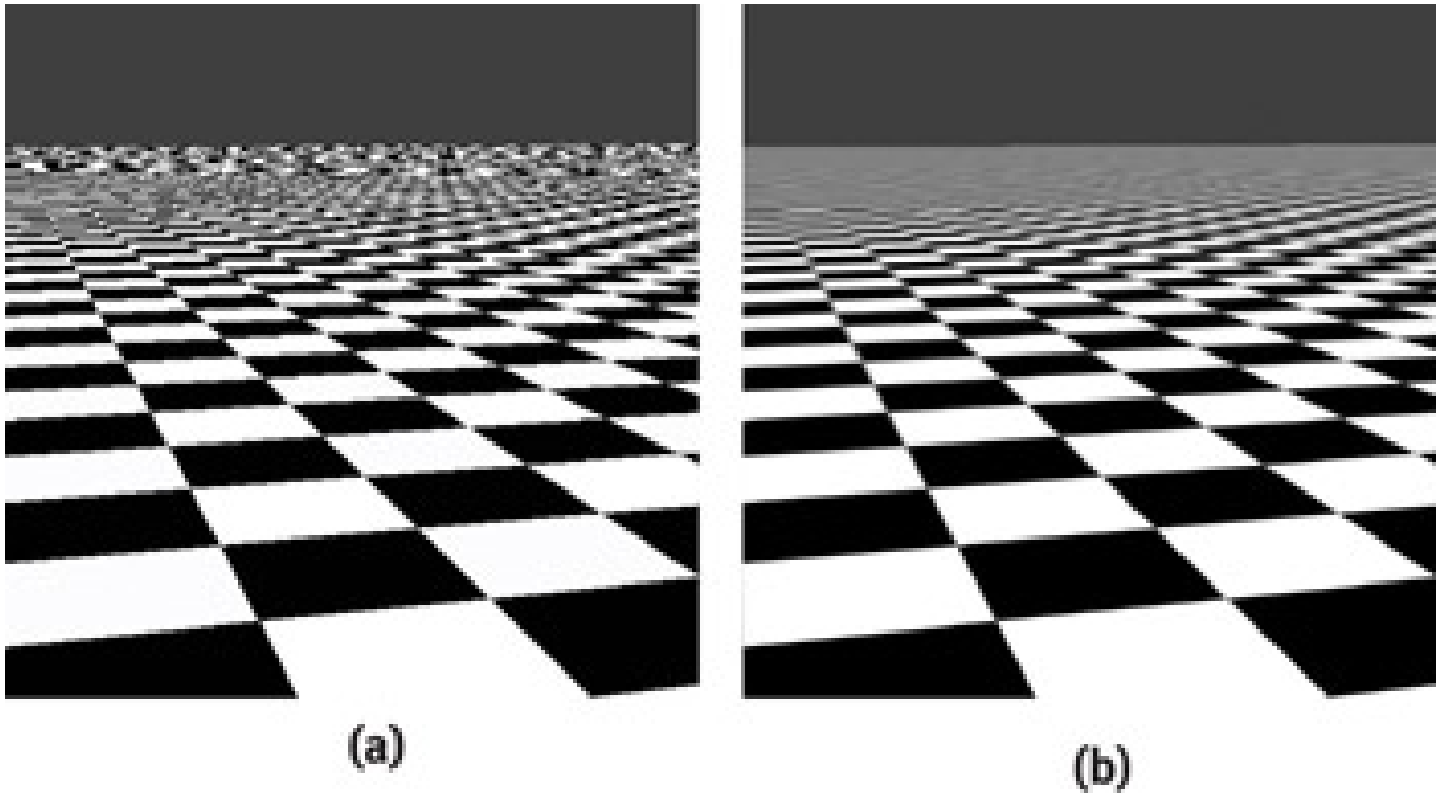

D60
Artifact not detail
SD9


Artifact not detail
SD9

Aliasing in Action: Graphics Example

Left: High-frequency checkerboard pattern gives aliasing patterns when sampled onto few pixels (in the distance).

Right: High frequencies are filtered/blurred out, to avoid aliasing.



Aliasing in Action: Stroboscopic Effect

Temporal aliasing of video data is responsible for the “stroboscopic effect” and the “wagon-wheel effect”.

“Slow-motion” falling droplets:

- http://www.youtube.com/watch?v=OtxlQTmx1LE

Weird suspended water stream:

- http://www.youtube.com/watch?v=uENITui5_jU

Wagon-wheel effect on car tire :

- https://www.youtube.com/watch?v=Ce_jRfM9b4k

Helicopter taking off with stationary rotor:

- https://www.youtube.com/watch?v=yr3ngmRuGUc

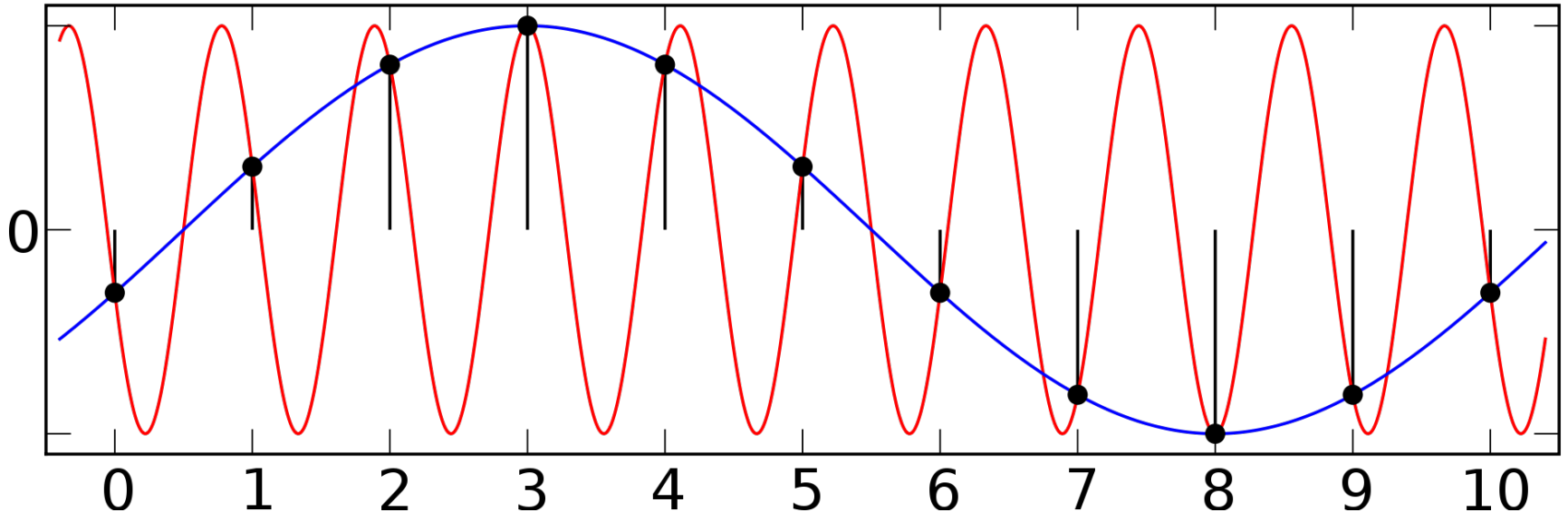
“3D Zoetrope”:

- https://www.youtube.com/watch?v=3-rPn0a56WE

Aliasing: The Underlying Mathemagic

Let's take a look at the mathematical cause of aliasing.

We will relate our N approximate Fourier coefficients F_k to the infinitely many continuous coefficients c_k.



Nyquist frequency

$\frac{N}{2T}$ is called the *Nyquist frequency.*

If the original continuous signal has non-zero frequencies such that

$$|k| > \frac{N}{2}$$

those frequencies “alias” (i.e., get added) to a lower frequency.

Once a signal is discretely sampled, there is no way to distinguish aliased frequencies.

Result: We have to sample at **twice** the rate of the highest occurring frequency in the original signal if we want to avoid aliasing.

Treating Aliasing

Two (partial) solutions:

(1) Increase the sampling rate/resolution to *capture* higher frequencies.

(2) Filter *before* sampling (digitizing) to *remove* (too) high frequencies.

Example of (2):

Digital cameras often have “optical lowpass filters” or “anti-aliasing filters” that physically blur the incoming light, *before* it hits the image sensor.