Image Compression

EE274, Fall22

Slide resources

- *“The Unreasonable Effectiveness of JPEG: A Signal Processing Approach” (Youtube video -> Reducible, beautiful illustrations!)* *https://www.youtube.com/watch?v=0me3guauqOU*

- *EE398A Stanford Lecture Notes: (Bernd Girod)* *https://web.stanford.edu/class/ee398a/handouts*

What is an image?

What is an image?

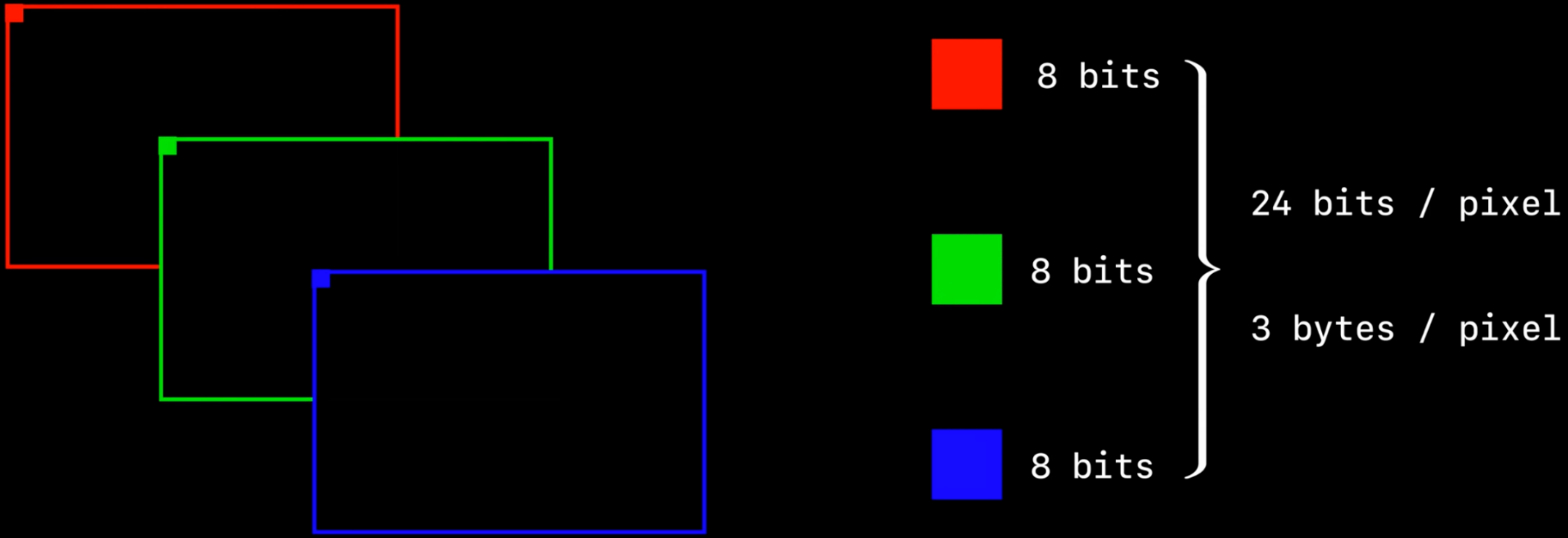


Image Compression

764x512x3 bytes
= 1.1MB!
(Uncompressed)

Image from Kodak dataset

Image Compression -> JPEG 40x

Image from Kodak dataset

Uncompressed -> 1.1MB
JPEG -> 27KB (~40x!)

Image Compression -> JPEG 80x

Image from Kodak dataset

Uncompressed -> 1.1MB
JPEG -> 14KB (~80x!)

Image Compression -> JPEG 137x

Image from Kodak dataset

Uncompressed -> 1.1MB
JPEG -> 8KB (~137x!)

Image Compression -> BPG

Uncompressed -> 1.1MB
BPG -> 8KB (~137x!)

Image from Kodak dataset

HiFiC -> ML-based image compression

Uncompressed -> 1.1MB
BPG -> 8KB (~137x!)

Image from Kodak dataset

Lossy Compression

- Incredible performance gains! ~40x-137x gains without much noticeable difference (depending upon the codec)

- So ubiquitous, my DSLR camera does JPEG compression by default :-| .. (difficult to find a “dataset” of non-compressed images)

- JPEG, JPEG2000, BPG (HEIC), AVIF, JPEG-XL, ML-based image compressors …

JPEG Image Compression

IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, FEBRUARY 1992

THE JPEG STILL PICTURE COMPRESSION STANDARD

Gregory K. Wallace
Multimedia Engineering
Digital Equipment Corporation
Maynard, Massachusetts

commonplace in general-purpose computing systems the way text and geometric graphics are. The majority of modern business and consumer usage of photographs and other types of images takes place through more traditional analog means.

The key obstacle for many applications is the vast amount of data required to represent a digital image

directly. A digitized version of a single, color picture at TV resolution contains on the order of one million bytes; 35mm resolution requires ten times that amount. Use of digital images often is not viable due to high storage or transmission costs, even when image capture and display devices are quite affordable.

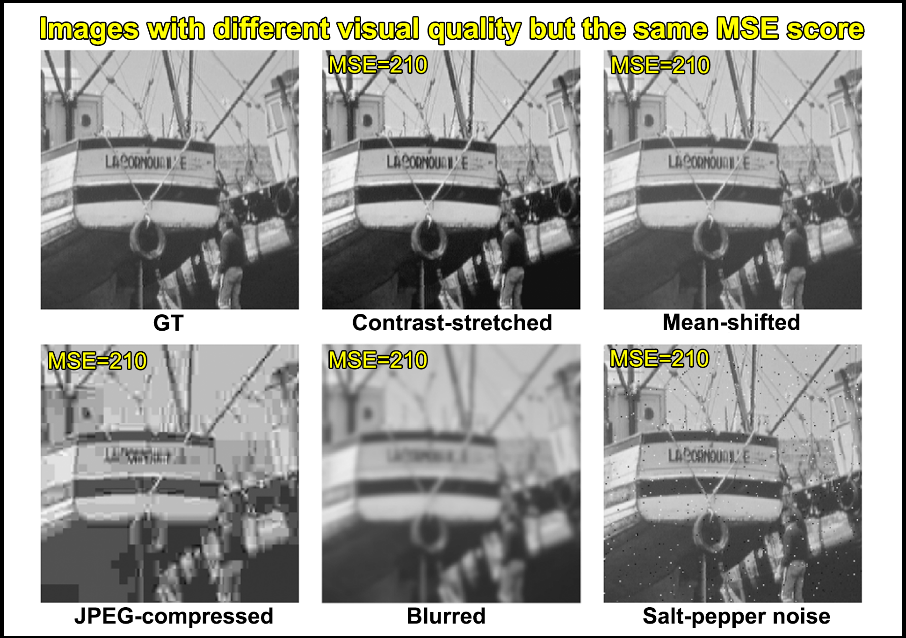
Lossy Compression -> Problem definition

Distortion metric -> MSE?

Lossy Compression -> Problem definition

Distortion metric -> ~~MSE~~?

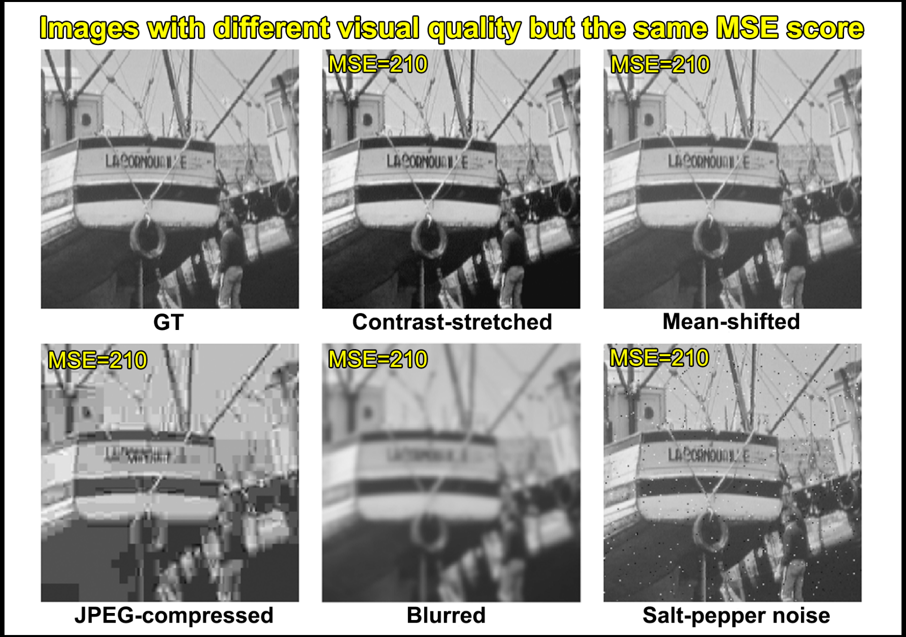
Lossy Compression -> Problem definition



Lots of research into understanding "Human Perceptual loss"...

Distortion metric -> ~~MSE~~?

Lossy Compression -> Problem definition



Lots of research into understanding “Human Perceptual loss”...

For simplicity, we will consider MSE (+heuristics)

Distortion metric -> MSE?

Lossy Compression -> Problem caveats

- Typically we care about compressing a single image, and not a *group* of images

- Non-asymptotic performance of various techniques is important

- Data is most likely non-stationary -> (need to convert/transform appropriately)

Lossy Compression -> Tools we know

- Scalar Quantization: fast, not the best
- Vector Quantization: very good, but slow as dim increases
- Transform Coding: Decorrelate data, and then use simpler (scalar) quantization
- Predictive coding: Fancier delta coding

All are useful in different contexts!

Vector Quantization / Color Quantization

Original Image

Simple Idea ->
quantize the 3-dim vector (R,G,B)

Vector Quantization / Color Quantization

Simple Idea ->
quantize the 3-dim vector
(R,G,B)

Number of colors = 256
(3x compression!)

Vector Quantization / Color Quantization

Simple Idea ->
quantize the 3-dim vector (R,G,B)

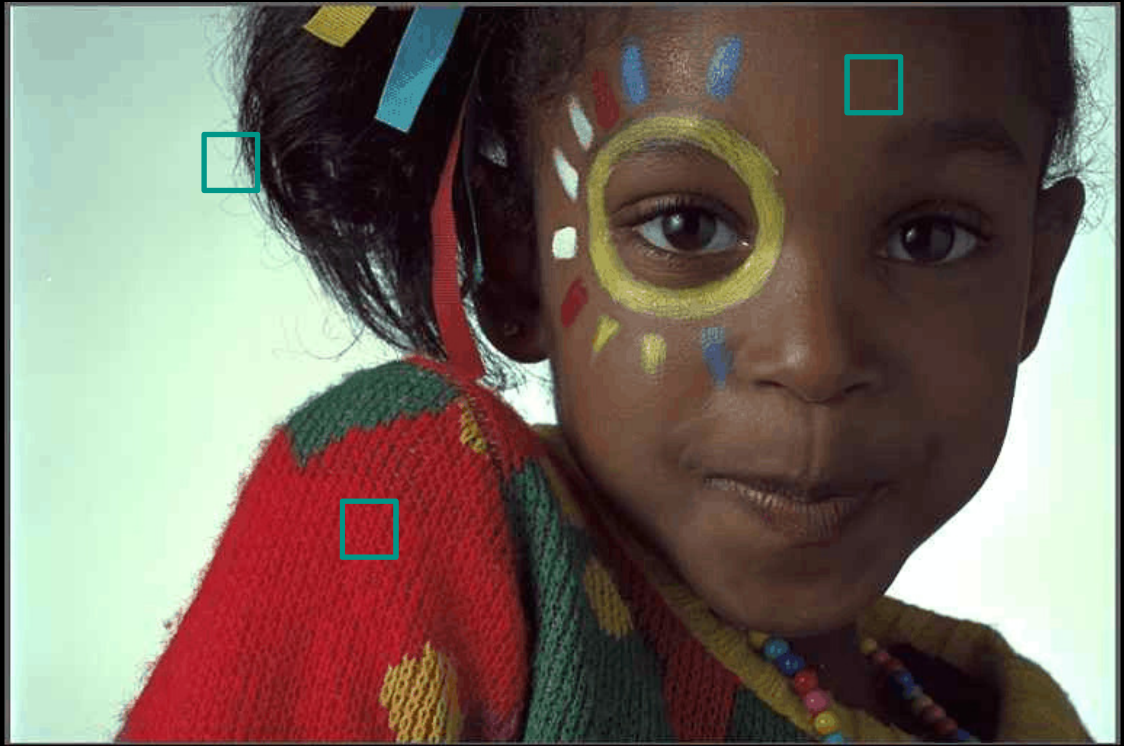
Number of colors = 16
(6x compression!)

Vector Quantization / Color Quantization

Number of colors = 256
(3x compression!)

***Q:** How can we further improve compression?*

Vector Quantization / Color Quantization



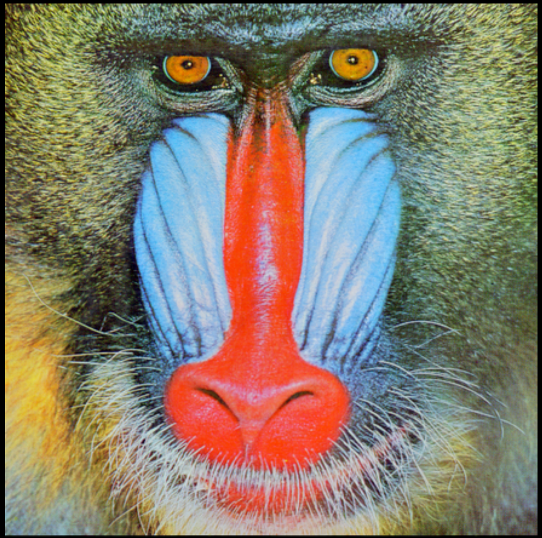
Vector Quantization / Color Quantization

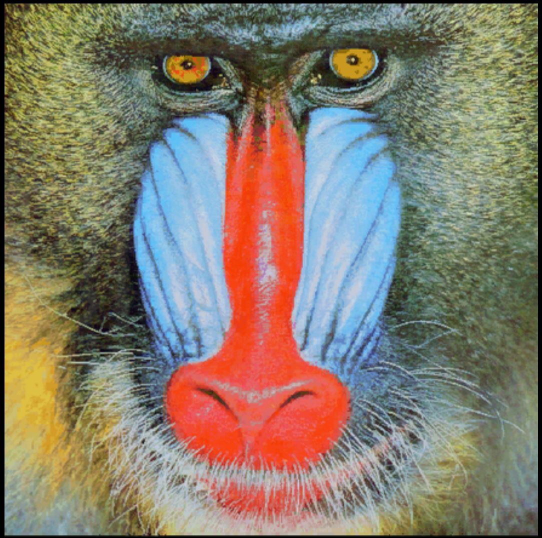
Number of colors = 256
(3x compression!)

***Q:** How can we further improve compression?*

Ans: Exploit "correlation" between neighboring pixels

Color Cell Compression





Use *Correlation* between neighboring pixels

Color Cell Compression (1984) -> use 2 colors (among 256) for each 4x4 block.
Effective -> 2 bits/pixel = 12x compression!

Color Cell Compression

Use *Correlation* between neighboring pixels

Color Cell Compression (1984) -> use 2 colors (among 256) for each 4x4 block.
Effective -> 2 bits/pixel = 12x compression!

Color Cell Compression

Use *Correlation* between neighboring pixels

Color Cell Compression (1984) -> use 2 colors (among 256) for each 4x4 block.
Effective -> 2 bits/pixel = 12x compression!

Exploiting Spatial correlation in the data

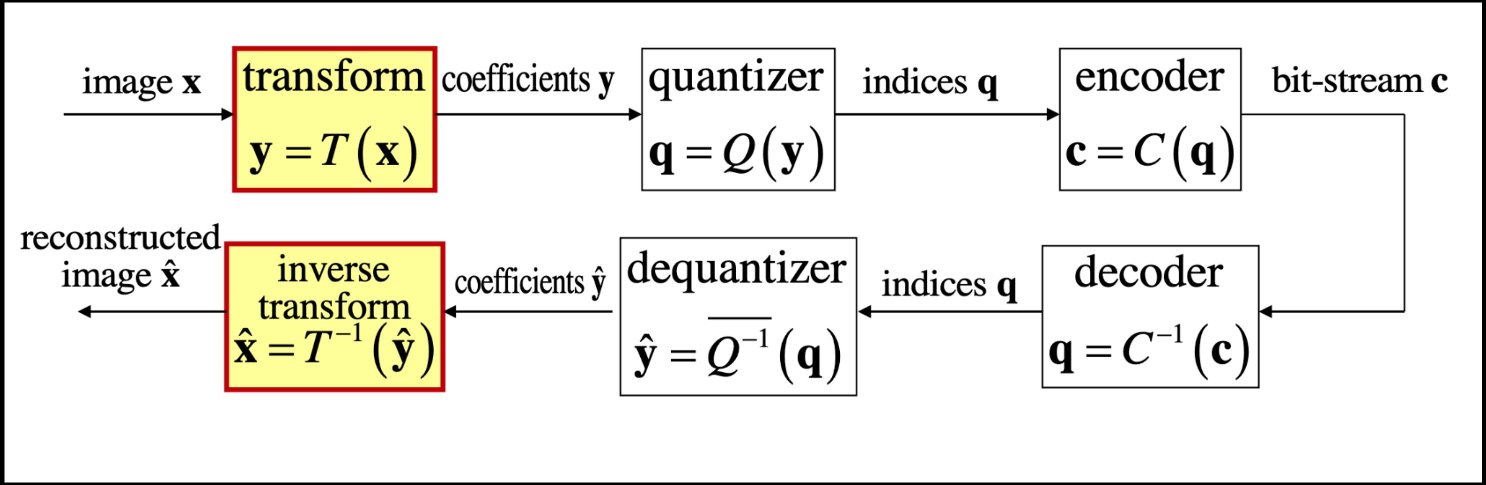
Key Idea -> We need to somehow exploit/remove the correlation between neighboring pixels.

Exploiting Spatial correlation in the data

Key Idea -> We need to somehow exploit/remove the correlation between neighboring pixels.

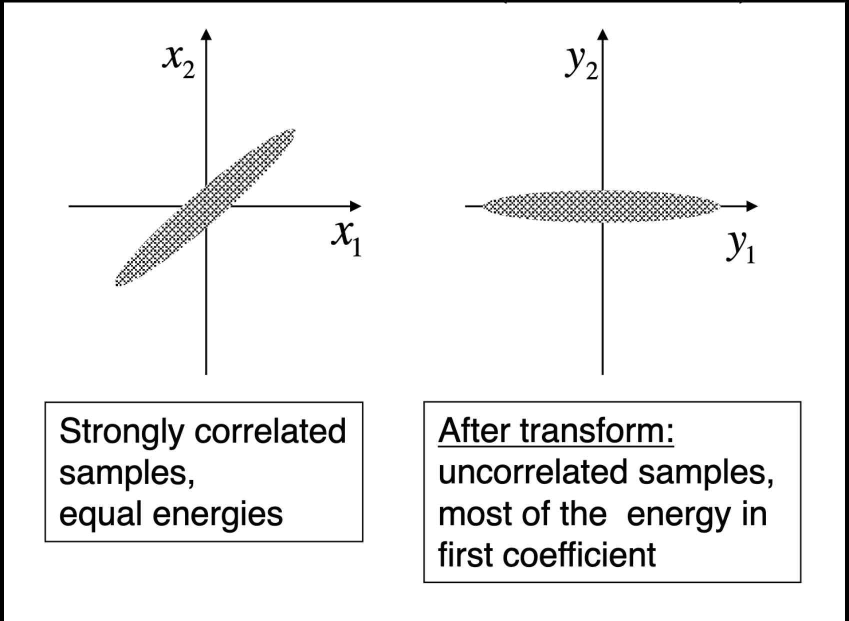
TRANSFORM CODING!

Transform Coding -> RECAP



Linear Transform Coding -> Example

Orthonormal transform -> "Rotate" the data appropriately so that the components are un-correlated



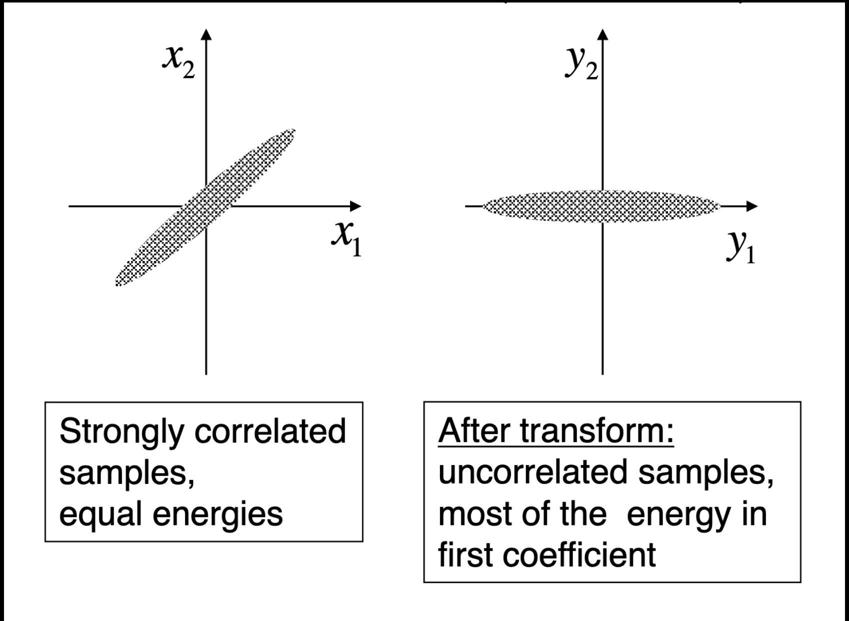
$$T((X_1, X_2)) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

$$Y_1 = \frac{X_1 + X_2}{\sqrt{2}}$$

$$Y_2 = \frac{X_1 - X_2}{\sqrt{2}}$$

Linear Transform Coding -> Example

Orthonormal transform -> "Rotate" the data appropriately so that the components are un-correlated



$$Y_1 = \frac{X_1 + X_2}{\sqrt{2}}$$

$$Y_2 = \frac{X_1 - X_2}{\sqrt{2}}$$

$$\mathbb{E}[X_1^2] = 1, \qquad \mathbb{E}[X_2^2] = 1$$

$$\mathbb{E}[Y_1^2] = 1.95, \qquad \mathbb{E}[Y_2^2] = 0.05$$

$$\mathbb{E}[X_1 X_2] = 0.97, \qquad \mathbb{E}[Y_1 Y_2] = 0$$

"energy compaction"

Transform Coding TL;DR

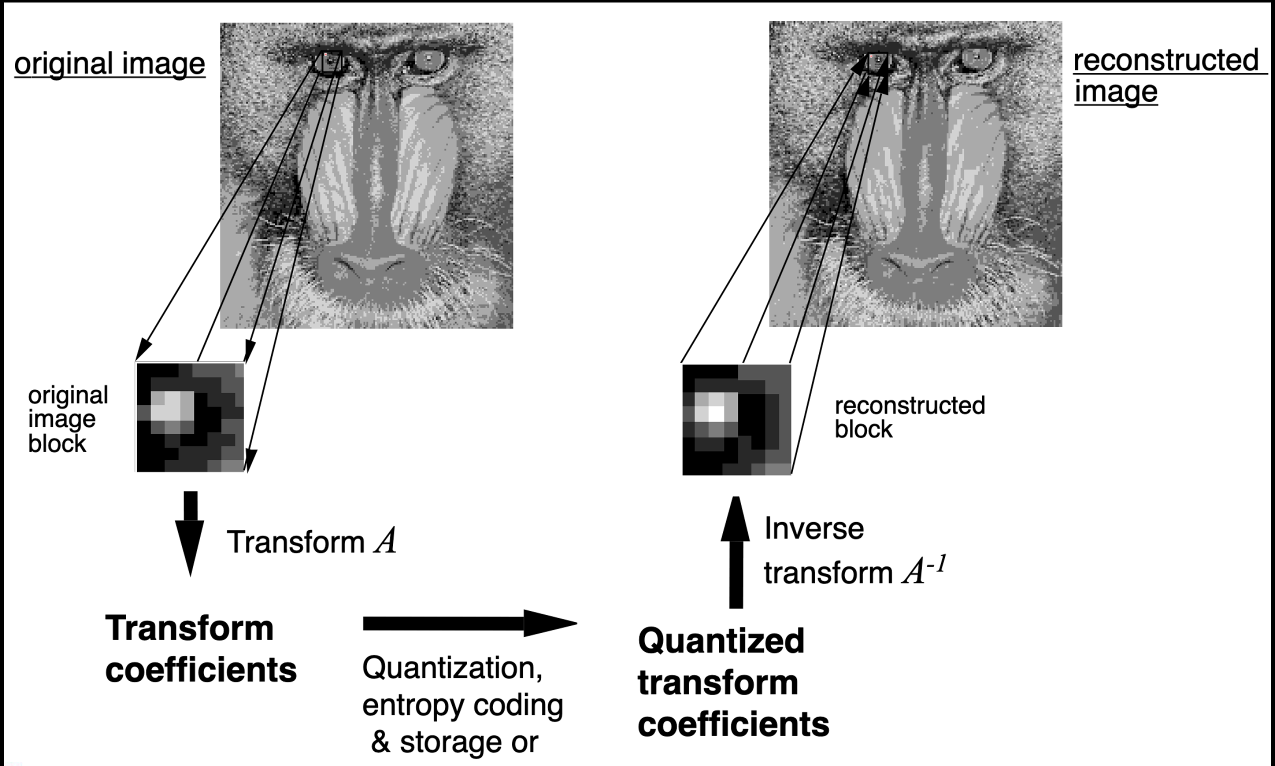
- Transform input data X -> Y, such that components of Y are uncorrelated.

$$\mathbb{E}[X_1^2] = 1, \qquad \mathbb{E}[X_2^2] = 1$$
$$\mathbb{E}[Y_1^2] = 1.95, \qquad \mathbb{E}[Y_2^2] = 0.05$$
$$\mathbb{E}[X_1 X_2] = 0.97, \qquad \mathbb{E}[Y_1 Y_2] = 0$$

- If components of X are indeed highly correlated, then components of Y will have “skewed energies” (one component has high variance, and others have lower variance)

- Can focus quantization efforts on Y1, and set Y2=0

Block Transform Coding



Linear Transform Coding

- Forward transform

$$\boxed{\mathbf{y} = \mathbf{A}\mathbf{x}}$$

NxN transform coefficients, arranged as a column vector

Transform matrix of size $N^2 x N^2$

Image block of size *NxN*, arranged as a column vector

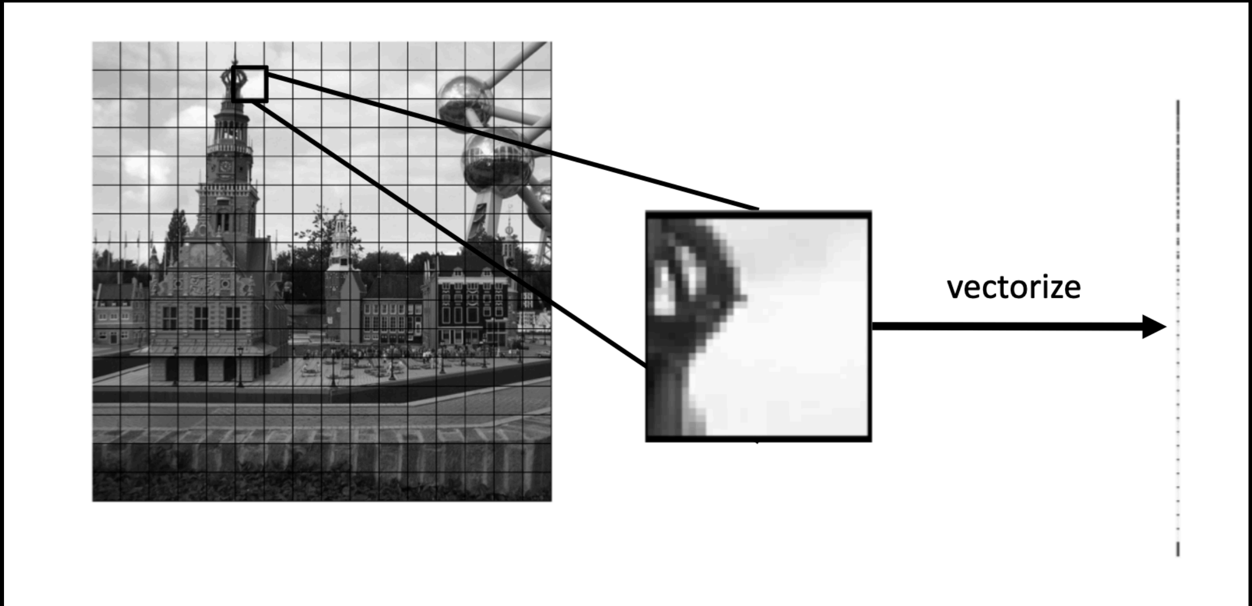
- Inverse transform

$$\boxed{\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} = \mathbf{A}^{T}\mathbf{y}}$$

Block Transform Coding

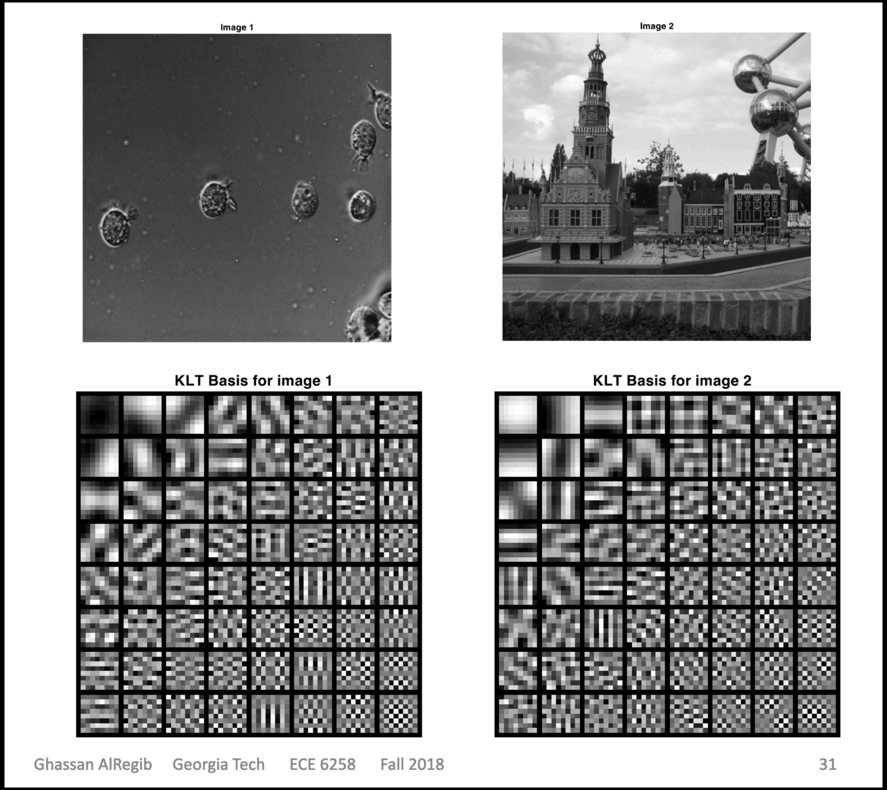
Step 1 -> Cut the image into blocks (eg 8x8), [grayscale]

X



KLT -> Transform Coding

Step 1 -> Cut the image into blocks **X**(eg 8x8)
Step 2 -> Find the transform matrix A
using Karhunen-Loeve Transform (KLT)



KLT -> Transform Coding

- Decorrelation by design: Decorrelated transform coefficients
- Depends upon the data: Transform depends upon the input image
- Slow: Non-structured matrix of size NxN = 64x64, matrix multiplication is N^2 (too slow :(), KLT construction is also slow

***Q:** Can we design a structured transform, which is close to optimal?*
(i.e. to the KLT matrix)

Transform Coding -> DCT

Q: *Can we design a structured transform, which is close to optimal?*
(i.e. to the KLT matrix)

How I Came Up with the Discrete Cosine Transform

Nasir Ahmed

Electrical and Computer Engineering Department, University of New Mexico, Albuquerque, New Mexico 87131

Transform Coding -> DCT

***Q:** Can we design a structured transform, which is close to optimal?*
(i.e. to the KLT matrix)

> What intrigued me was that the KLT was indeed the optimal transform on the basis of the mean-square-error criterion and the first-order Markov process model, and yet there was no efficient algorithm available to compute it. As such, the focus of my research was to determine whether it would be possible to come up with a good approximation to the KLT that could be computed efficiently. An approach that I thought might be worth looking into was *Chebyshev interpolation*, a neat discussion of which was available

Transform Coding -> DCT

***Q:** Can we design a structured transform, which is close to optimal?*
(i.e. to the KLT matrix)

Much to my disappointment, NSF did not fund the proposal; I recall one reviewer's comment to the effect that the whole idea seemed "too simple." Hence I decided to work on this problem with my Ph.D. student Mr. T. Natarajan and my friend Dr. Ram Mohan Rao at the University of Texas at Arlington. In fact, I re-