Data Compression with Linear Algebra

Chris and David Etler

Overview

- This slide-deck explains the process of image compression by presenting and explaining the key ideas involved
- Contents:
  - **Change of Basis** how a change of basis can be used to represent matrices more efficiently
  - **Entropy** how we can quantitatively measure information
  - **Lossy and Lossless Compression** the different kinds of compression and their methods, with examples
  - **Nature of Images** how images can be represented as matrices, and how compression artifacts affect the image
  - **Examples** of systems compressed via a discrete cosine transform

Change of Basis and JPEG

- Change of basis is a transformation from one coordinate system to another
  - E.G., from Cartesian to polar coordinates
- Certain bases can convey the same system with less information
  - This works by making certain coefficients less significant, without reducing the number of dimensions (lossless compression)
  - We can go further and set insignificant coefficients to zero (lossy compression)
- In the context of image compression, the *spacial domain* means "standard basis" and *frequency domain* means "cosine basis"
  - In general, the frequency domain could be represented by a different basis (ie Wavelet basis in JPEG2000)

Discrete Cosine Matrix

- A basis for R^8 vectors can be encoded as row vectors in the *discrete cosine matrix*, D_8

$$D_8 = \frac{1}{2}\begin{pmatrix}
\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\
\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{11\pi}{16} & \cos\frac{13\pi}{16} & \cos\frac{15\pi}{16} \\
\cos\frac{2\pi}{16} & \cos\frac{6\pi}{16} & \cos\frac{10\pi}{16} & \cos\frac{14\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{22\pi}{16} & \cos\frac{26\pi}{16} & \cos\frac{30\pi}{16} \\
\cos\frac{3\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{27\pi}{16} & \cos\frac{33\pi}{16} & \cos\frac{39\pi}{16} & \cos\frac{45\pi}{16} \\
\cos\frac{4\pi}{16} & \cos\frac{12\pi}{16} & \cos\frac{20\pi}{16} & \cos\frac{28\pi}{16} & \cos\frac{36\pi}{16} & \cos\frac{44\pi}{16} & \cos\frac{52\pi}{16} & \cos\frac{60\pi}{16} \\
\cos\frac{5\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{25\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{45\pi}{16} & \cos\frac{55\pi}{16} & \cos\frac{65\pi}{16} & \cos\frac{75\pi}{16} \\
\cos\frac{6\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{30\pi}{16} & \cos\frac{42\pi}{16} & \cos\frac{54\pi}{16} & \cos\frac{66\pi}{16} & \cos\frac{78\pi}{16} & \cos\frac{90\pi}{16} \\
\cos\frac{7\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{49\pi}{16} & \cos\frac{63\pi}{16} & \cos\frac{77\pi}{16} & \cos\frac{91\pi}{16} & \cos\frac{105\pi}{16}
\end{pmatrix}$$

Figure 1: The 8x8 DCT Matrix used to calculate the discrete cosine transform

Discrete Cosine Matrix

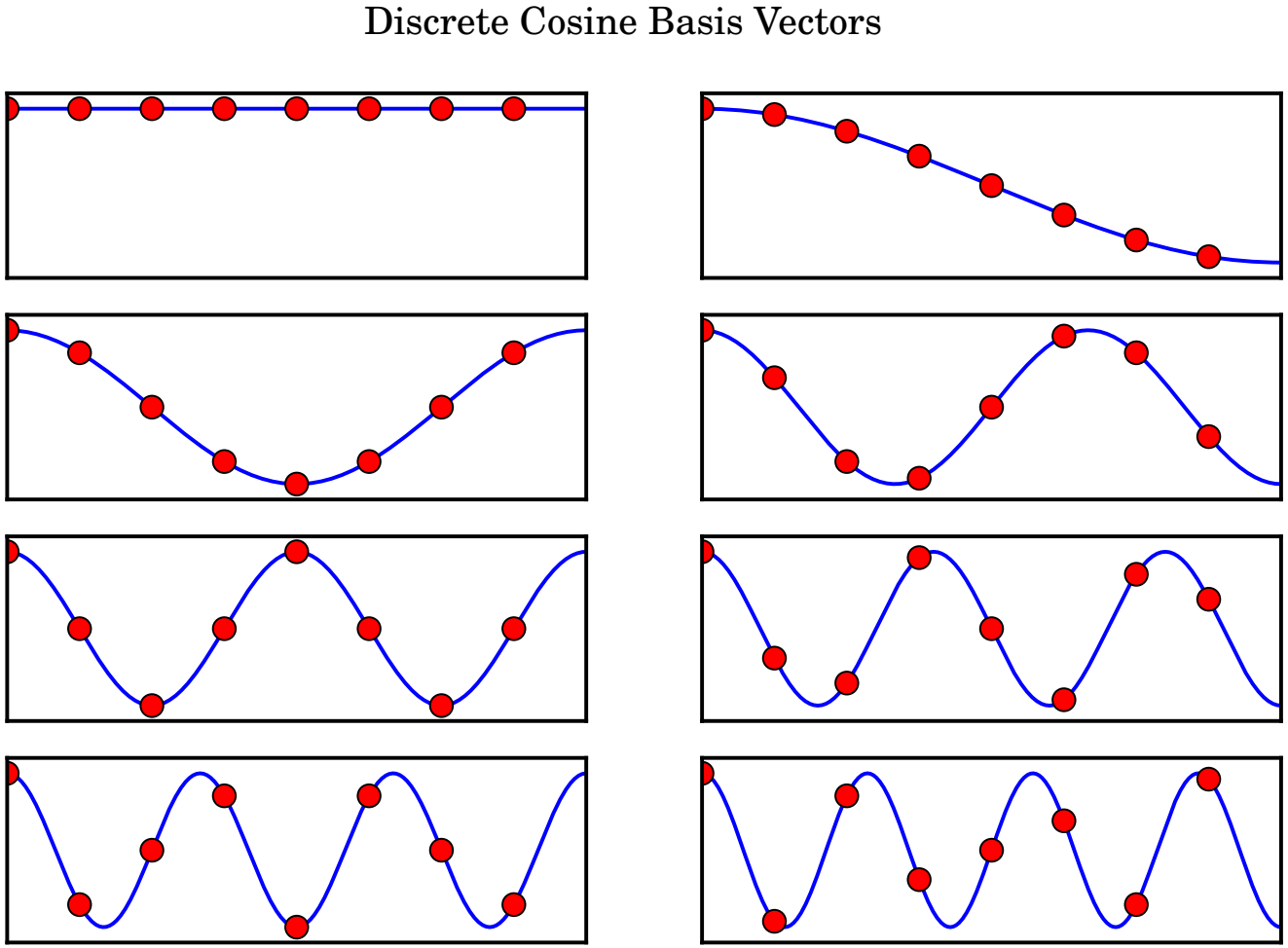

Figure 2: The 8 orthonormal vectors form a basis for R^8

Discrete Cosine Transform

- Change of basis from standard basis to the cosine basis
  - In image compression, an 8x8 matrix representing pixels in space is mapped to an 8x8 matrix representing pixels as a superposition of cosines
- When used on real-world photography, many of a the coefficients w.r.t the cosine basis are nearly 0
  - This is bescause all rows in D_8 after the the first (called the *direct current* (DC) vector) sum to zero
  - Therefor, an image with many rows of similar pixels with have many coefficients near 0 w.r.t the cosine basis
  - These coefficients can be ignored without ruining image quality
- In terms of the cosine matrix, the discrete cosine transform of a column vector $\vec{v}$ is given as $\mathrm{DCT}(\vec{v}) = D\vec{v}$
- The DCT of a row vector $\vec{v}_{row}$ is given as $\mathrm{DCT}(\vec{v}_{row}) = \vec{v}_{row} D^T$

Discrete Cosine Transform in 2D

- The discrete cosine transform matrix D maps a single column vector to the cosine basis
  - When applied to a matrix M, DM maps each column vector of M to its representation in the cosine basis
- When compressing images, we want to transform both the rows and the columns, which can be computed as such:
  1. Take the DCT of every column vector by multiplying by D to get DM
  2. Take the DCT of every row vector in the result by computing $(DM)D^T = DMD^T = \mathrm{DCT}(M)$
- The same result can be obtained reversing the order: $D(MD^T) = DMD^T$
- The inverse discrete cosine transform of a matrix M' is given as $\mathrm{IDCT}(M') = D^T M' D$ (because the vectors of D are orthonormal)

Discrete Cosine Transform in 2D

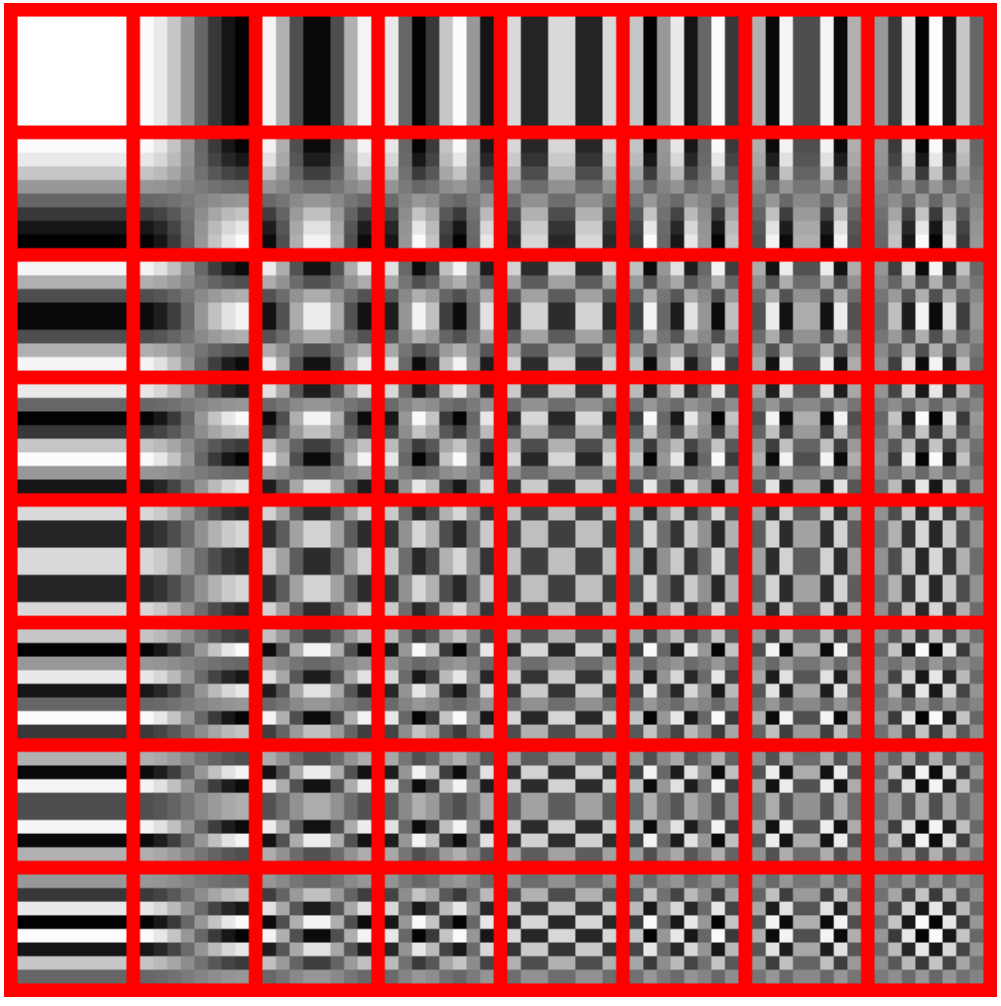

Figure 3: The 64 (8x8) basis matrices associated with the 2D DCT of length 8 form a basis for M_{8x8}

Image Recomposition

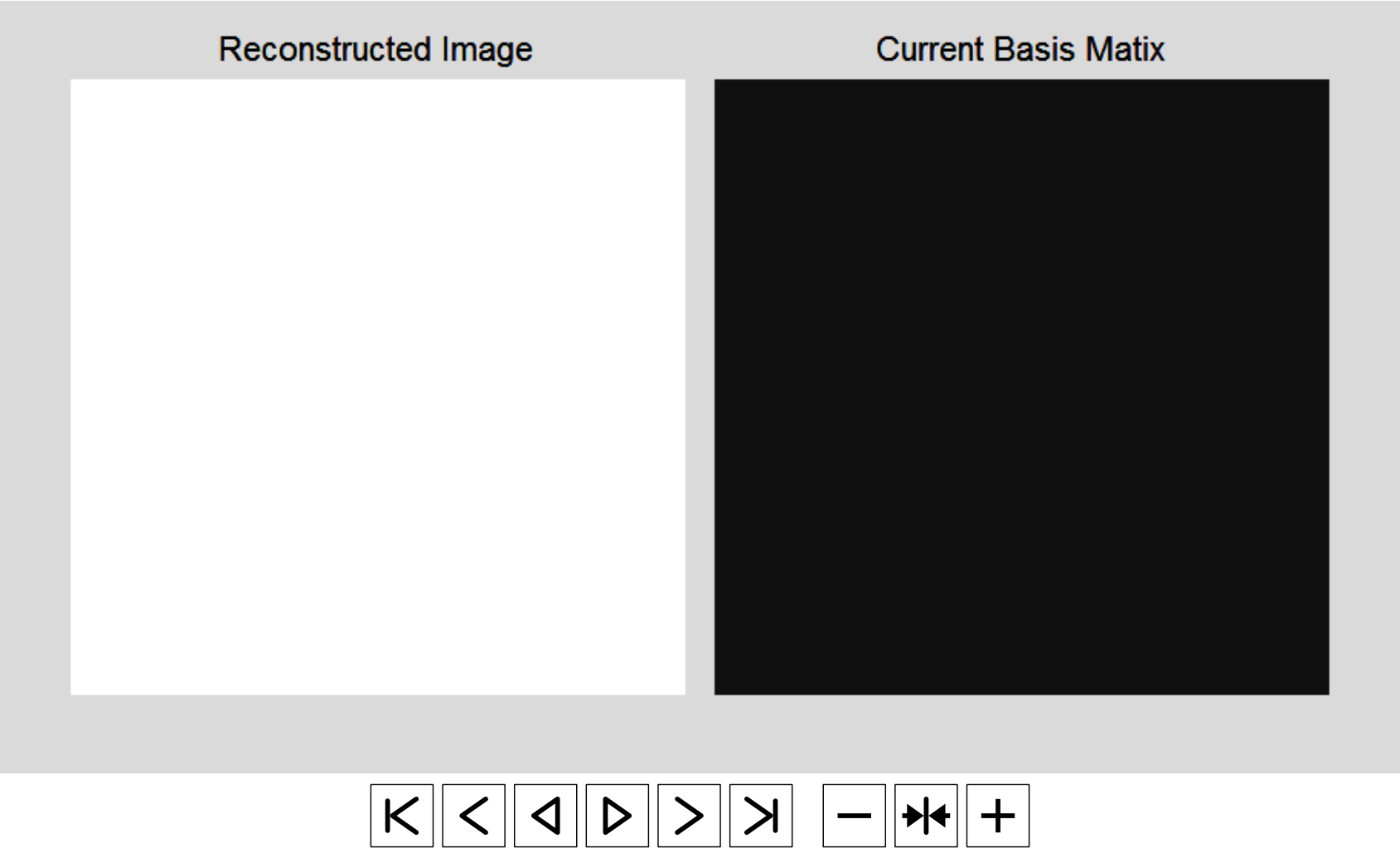

Figure 4: An animation showing the re-composition of an image from a linear combination of 256 16x16 basis vectors for M_{16x16}

Entropy

- Entropy is a formal framework for quantitatively describing the amount of information encoded in a system
- Discovered/formalized by Claude Shannon, founder of Information Theory
- Entropy of a system X which has states x_i occurring with a probability of $P(x_i)$ is given by the equation:

$$\text{Entropy}(X) = -\sum_{i=1}^{N} P(x_i) \log_2 P(x_i)$$

- When all N possible states of x are likely this simplifies to

$$\text{Entropy}(\{x_1, \cdots, x_N\}) = \log_2(N)$$

- Useful framework for discussing the theory of information
  - In particular, we can compare the efficiency of different compression algorithms by comparing the relative amounts of entropy

Entropy

Table 1: A few examples of entropy for various systems

System	Entropy
Outcome of N Fair Coin Tosses	N bits
A Single Word in the English Language	11 bits
A System with a Single Possible State	0 bits (no information)

Lossless Compression

- Re-encode information by performing a *change of basis*
- Changing back to the *standard basis* completely recovers the original system
- Entropy can be reduced by changing to a basis which a lower number of probable states
  - File size is directly dependent on entropy
  - File size also depends on *encoding scheme*. For the simplicity's sake, we assume 100% efficient encoding
  - ie, we assume the file size is equal to the entropy of the system
- Eigenvector basis is the theoretically most efficient basis, but would take far too much computational power to determine for even a moderately-sized system

Lossy Compression

- Extends lossless compression by also removing information from the system
- A typical lossly compression algorithm might work as follows to encode a system:
  1. Perform a change of basis
  2. Remove information from the system
     - via an algorithm such as *quantization* or *thresholding*
  3. Encode the system relative to the new basis, as well as information about the chosen basis
- The decoding process would then work as follows:
  1. Use the information about the new basis to change back to the standard basis
  2. The new system in the standard basis should closely resemble the original system, but some artifacts will be present due to the removal of entropy from the system

Quantization

- Quantization works by directly limiting the number of possible states a system can take on
- As such, it increases the probability of each remaining state, lowering the total entropy
- For example, we may limit a system that can take on reals to a system that can take on integers
  - eg, limit $x_i \in [-5, 5]$ to $x_i' \in \{-5, -4, \cdots, 4, 5\}$
- Such a system would look similar the original, but with clear artifacts

Quantization – Example

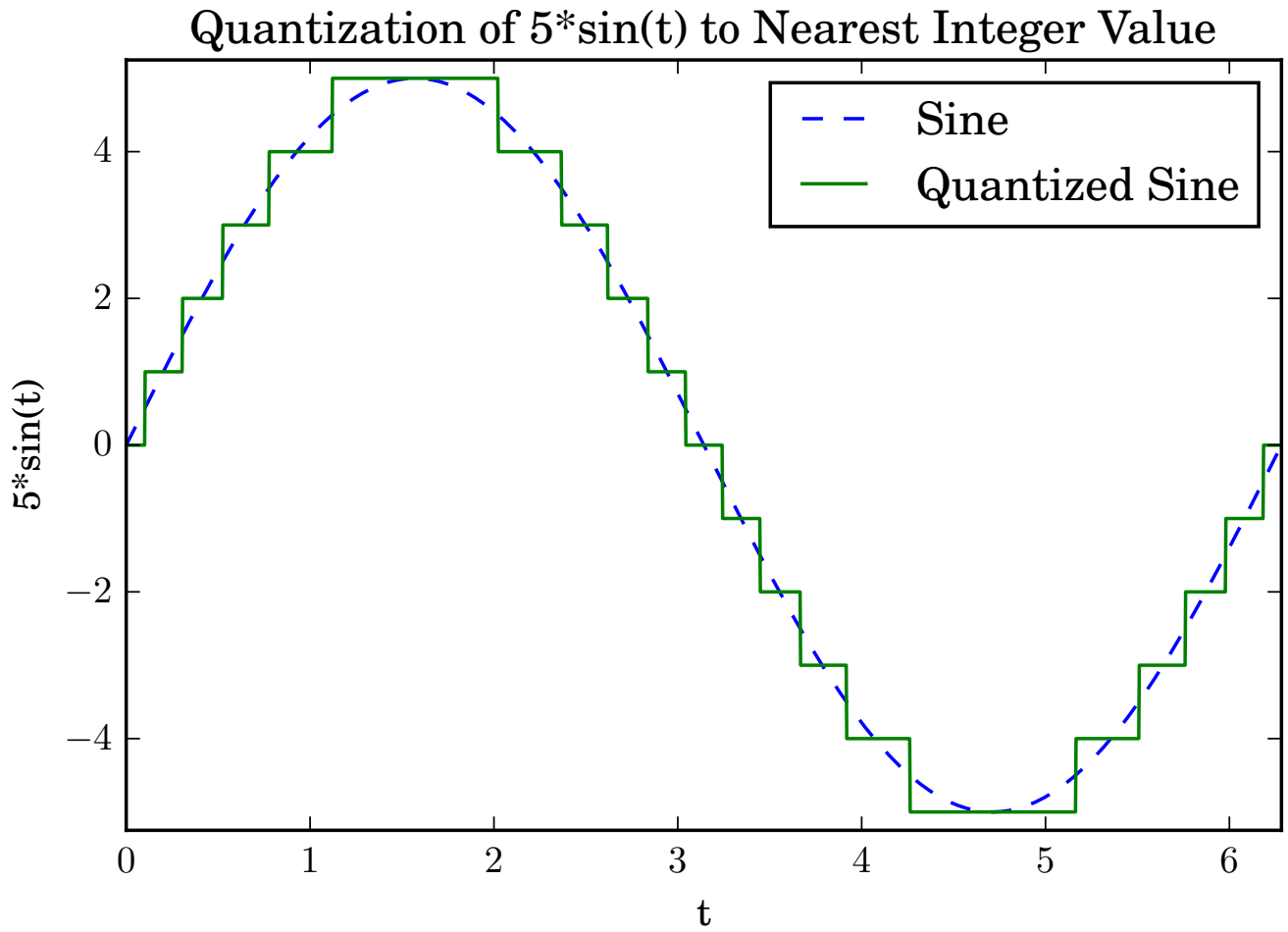

Figure 5: A system quantized in the standard basis can show obvious artifacts

Lossy Compression — Choice of Basis

- We want to remove as much information from the system as possible while introducing the least amount of noticeable artifacts
- The standard basis is typically a poor choice because quntization and thresholding both introduce obvious artifacts
- The eigenvector basis would be the most efficient, but calculating it would be too costly
- One basis must be chosen as the default for a given algorithm, so that the choice of basis does not need to be encoded alongside the system
  - JPEG chooses the discrete cosine basis, which has been empirically shown to be particularly efficient at compressing photographic images

Nature of Images

- Images are made of up *pixels*, discrete spacial components holding a single color value
- We can represent a 2D grayscale image as a matrix of intensity values, called M, where each M_{ij} encodes one pixel
- A color image can be thought of as set of three matrices $I = \{R, G, B\}$ which encode the intensity of the red, blue, and green values respectively
  - These three components are *linearly independent* because no combination of blue and green can make red, and vice versa
  - Further, the three color components *span* the space of possible *hues*, *intensities*, and *saturations*
  - The pixels on a computer's monitor are divided into sub-pixels of red, green, and blue for this reason
  - These three matrices are referred to as the *channels* of the image

Example Image

Figure 6: Orginal Image (entropy per pixel: 24 bits)

Image Compression via Quantization

- By performing lossy compression on images, we can remove the details which humans are insensitive to while preserving the information which we are most sensitive to
- A common technique for lossy compression is *quantization*, reducing the number of possible values each pixel can take on
  - Related to a similar method called *thresholding*, where low-valued data are simply discarded (set to 0)
- This reduces the overall information in the image, but our eyes aren't *that* sensitive to small changes in color, right?

Image Compression via Quantization – Continued

- Typically pixels values are 8-bits per *channel* (ie, $2^8 = 256$ possible states per channel)
- We can halve the number of possible states by computing $I' = 2 * \text{floor}(I/2)$
- In practice, we could go as far as quantizing to $I' = 16 * \text{floor}(I/16)$
- This computation is performed three times, once for each channel
- Quantizing in the standard basis simply limits the possible values of each pixel in space

Example Image after Compression in the Spacial Domain

Figure 7: Image compressed w.r.t standard basis (entropy per pixel: 6 bits)

Orginal Image Reference

Compression in the Spacial Domain — Comments

- We can clearly see that we are, after all, sensitive to small changes in colors arranged spacially
  - The biggest issue with naiave spacial quantization is *gradients*, where one color smoothly fades into another
  - While we can't easily distinguish two similar colors, we can notice general trends in colors
  - By quantizing the possible color values for each pixel w.r.t the standard basis, we create *color bands*, which are distracting and visually unappealing
- A total of 18 bits were removed per pixel
  - This is due to the high amount of redundancy in the original image (not all colors were equally as likely)

Quantization in the Frequency Domain — Motivation

- Motivated by our high sensitivity to the relation between space and color, we decide to perform our compression with respect to a different basis
- The main issue with spacial quantization is neighboring pixels often exhibit a smooth change in color, which is poorly represented after quantization
  - Such a smooth change in color is called *low frequency* data
  - Spacial-color quantization best represents images with *high frequency* data: images where neighboring pixels often do not share similar colors
  - It turns out that humans are *most* sensitive to low frequency data and are relatively insensitive to high frequency data

Quantization in the Frequency Domain

- In order to remove the data humans are least sensitive to, we need to convert our image from spacial domain to frequency domain via a change of basis
  - Our image with respect to our new basis is computed via the discrete cosine transform
- After doing so, we can quantize the intensity values in the frequency domain exactly the same way as we did in the spacial domain, by limiting the possible set of values each element can take on
- Here we measure entropy in the frequency domain, rather than the entropy after the image is transformed back into the spacial domain
  - This is because the image is transmitted in this form, only to be transformed back later, a process called *decoding*

Example Image after Compression in Frequency Domain

Figure 8: Frequency Compressed Image (entropy per pixel: 5 bits)

Orginal Image Reference

Quantization in the Frequency Domain — Comments

- Though some compression artifacts are apparent, the image retains far more of its meaningful information than the spacially compressed one
- Here low frequency information is preserved whereas high frequency information is discarded
- This encodes the sky better, but some detail is lost in the grass
- Notice the overall image feels “blurry”, as all the sharp details are removed
- The actual entropy per bit was far lower than predicted, due to the quantization performing a thresholding on the low-frequency pixels
  - ie, the sky contained very little information, so a lot of information could be discarded

Frequency Domain Compression Without Quantinization

- Quantization does not need to be performed to compress the image
- Instead, the image need only be re-encoded in the frequency domain, and then transmitted directly
- This is a form of *lossless compression*
- Because our image has a lot of low frequency data, there is a lot of redundancy in the spacial domain by having to specify the color for every pixel
- This redundancy is removed when we transform to the frequency domain
- The entropy of our when encoded in the frequency domain was 7 bits per pixel

Image Compression — Takeaway

- Humans are more sensitive to low frequency information than they are to high frequency information
- Images often have higher redundancy when represented in the frequency domain rather than the spacial domain
- We can therefor use a change of basis to bring our image into the frequency domain, and then perform our compression there
- The image is transmitted as coefficients in the frequency domain, and then transformed back into spacial domain for viewing
- This process works best for photo-realistic images; frequency domain compression performs poorly on computer-drawn graphics with sharp boarders (high frequency content)

Summary of Compression Techniques

Table 2: Summary of compression techniques in different bases

Domain	Compression	Entropy per Pixel	Format
Spacial	Lossless	24 bits	Bitmap Image
	Lossy	6 bits	GIF
Frequency	Lossless	7 bits	Lossless JPEG
	Lossy	5 bits	Lossy JPEG

Quantization Example

- JPEG image compression is performed in 8x8 blocks
- Lets consider one 8x8 block I that represents a grayscale image with an 8-bit color channel

$$I = \begin{pmatrix} 92 & 199 & 222 & 62 & 210 & 174 & 249 & 254 \\ 206 & 109 & 57 & 145 & 119 & 40 & 168 & 131 \\ 137 & 22 & 34 & 126 & 166 & 109 & 203 & 212 \\ 42 & 169 & 31 & 150 & 18 & 165 & 14 & 218 \\ 209 & 122 & 168 & 64 & 124 & 169 & 184 & 44 \\ 161 & 51 & 1 & 78 & 228 & 82 & 13 & 232 \\ 78 & 56 & 29 & 64 & 119 & 181 & 197 & 183 \\ 152 & 16 & 71 & 185 & 113 & 26 & 86 & 195 \end{pmatrix}$$

Quantization Example

- Multiplying $D_8 I D_8^T$ where D_8 is the 8x8 DCT Matrix, we get:

$$\mathrm{DCT}(I) = \begin{pmatrix} 991 & -163 & 114 & 34 & 137 & -19 & -10 & 12 \\ 123 & 7 & 23 & -31 & -100 & -12 & -50 & 34 \\ 69 & -52 & 8 & -61 & 19 & 29 & -39 & 87 \\ 87 & 19 & -1 & 55 & -84 & -100 & 25 & 98 \\ 57 & 71 & -21 & -82 & -121 & -19 & 43 & -34 \\ 13 & -154 & 1 & -43 & -82 & -108 & 20 & -88 \\ 23 & -52 & -15 & -25 & 126 & -58 & 23 & 59 \\ 86 & -43 & -11 & 79 & -132 & 99 & -5 & 124 \end{pmatrix}$$

Quantization Example

- The same matrix, with 4-bits removed via quantization is:

$$Q = \begin{pmatrix} 976 & -176 & 112 & 32 & 128 & -32 & -16 & 0 \\ 112 & 0 & 16 & -32 & -112 & -16 & -64 & 32 \\ 64 & -64 & 0 & -64 & 16 & 16 & -48 & 80 \\ 80 & 16 & -16 & 48 & -96 & -112 & 16 & 96 \\ 48 & 64 & -32 & -96 & -128 & -32 & 32 & -48 \\ 0 & -160 & 0 & -48 & -96 & -112 & 16 & -96 \\ 16 & -64 & -16 & -32 & 112 & -64 & 16 & 48 \\ 80 & -48 & -16 & 64 & -144 & 96 & -16 & 112 \end{pmatrix}$$

Quantization Example

- Using the quantized matrix Q and reversing the transform via $I' = \mathrm{IDCT}(Q)$, we get the output:

$$I' = \begin{pmatrix} 32 & 218 & 204 & 58 & 203 & 175 & 246 & 248 \\ 223 & 100 & 55 & 140 & 130 & 39 & 169 & 129 \\ 127 & 23 & 26 & 121 & 151 & 111 & 209 & 213 \\ 45 & 164 & 27 & 144 & 12 & 166 & 10 & 216 \\ 196 & 129 & 168 & 70 & 121 & 165 & 179 & 43 \\ 161 & 49 & 1 & 70 & 229 & 87 & 12 & 232 \\ 76 & 55 & 27 & 61 & 115 & 182 & 193 & 183 \\ 147 & 16 & 63 & 187 & 113 & 29 & 80 & 206 \end{pmatrix} \approx I$$

Quantization Example

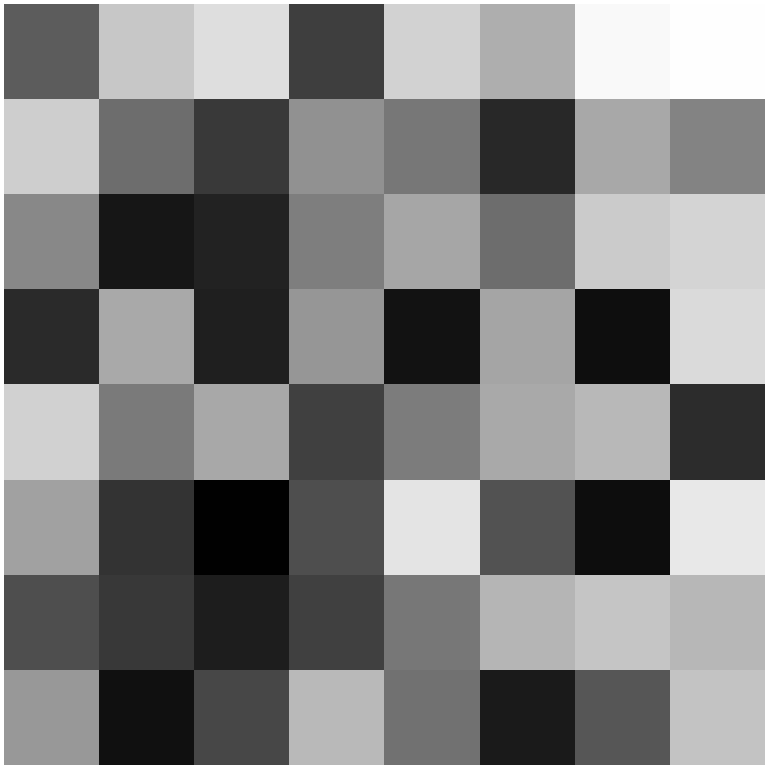

Figure 9: Original Image

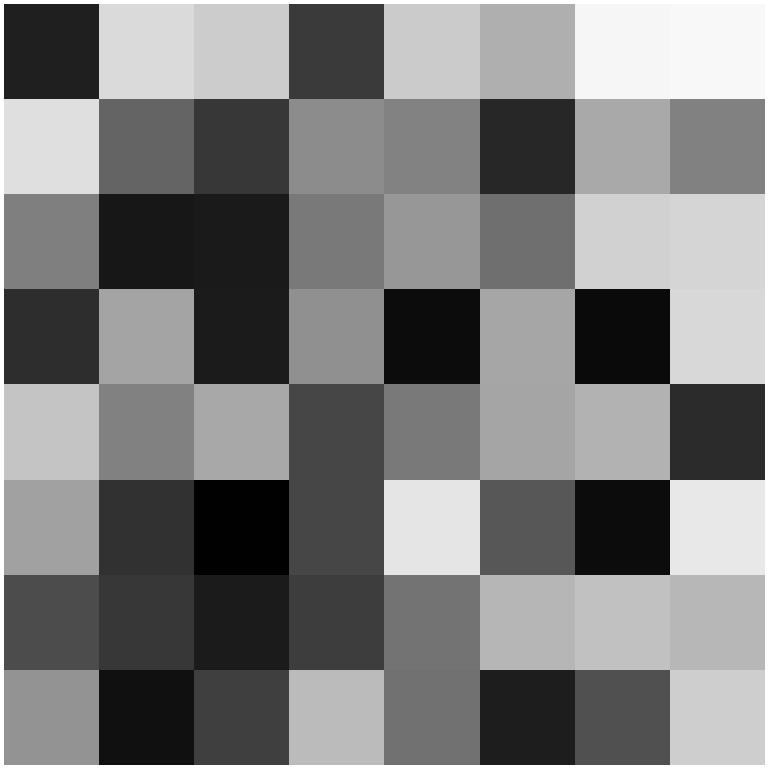

Figure 10: Compressed Image

- For a highly entropic image such as this, the DCT does little
- DCT keeps edges in tact whilst removing data from areas of repetition and similarity

Thresholding Example

- DCT works particularly well with blocks of consistent colors
- Consider this 4x4 matrix with very similar colors at each pixel:

$$I = \begin{pmatrix} 220 & 230 \\ 240 & 200 \end{pmatrix}$$

- After applying the DCT we have:

$$\mathrm{DCT}(I) = \begin{pmatrix} 445 & 15 \\ 5 & -25 \end{pmatrix} \approx \begin{pmatrix} 445 & 0 \\ 0 & 0 \end{pmatrix} = T$$

- Finally, we have

$$\mathrm{IDCT}(T) = \begin{pmatrix} 225 & 225 \\ 225 & 225 \end{pmatrix} \approx I$$

- Relative to the top element, the other elements are very low frequency and thus insignificant
- Elements can be set to 0 to save space without destroying image quality (thresholding)