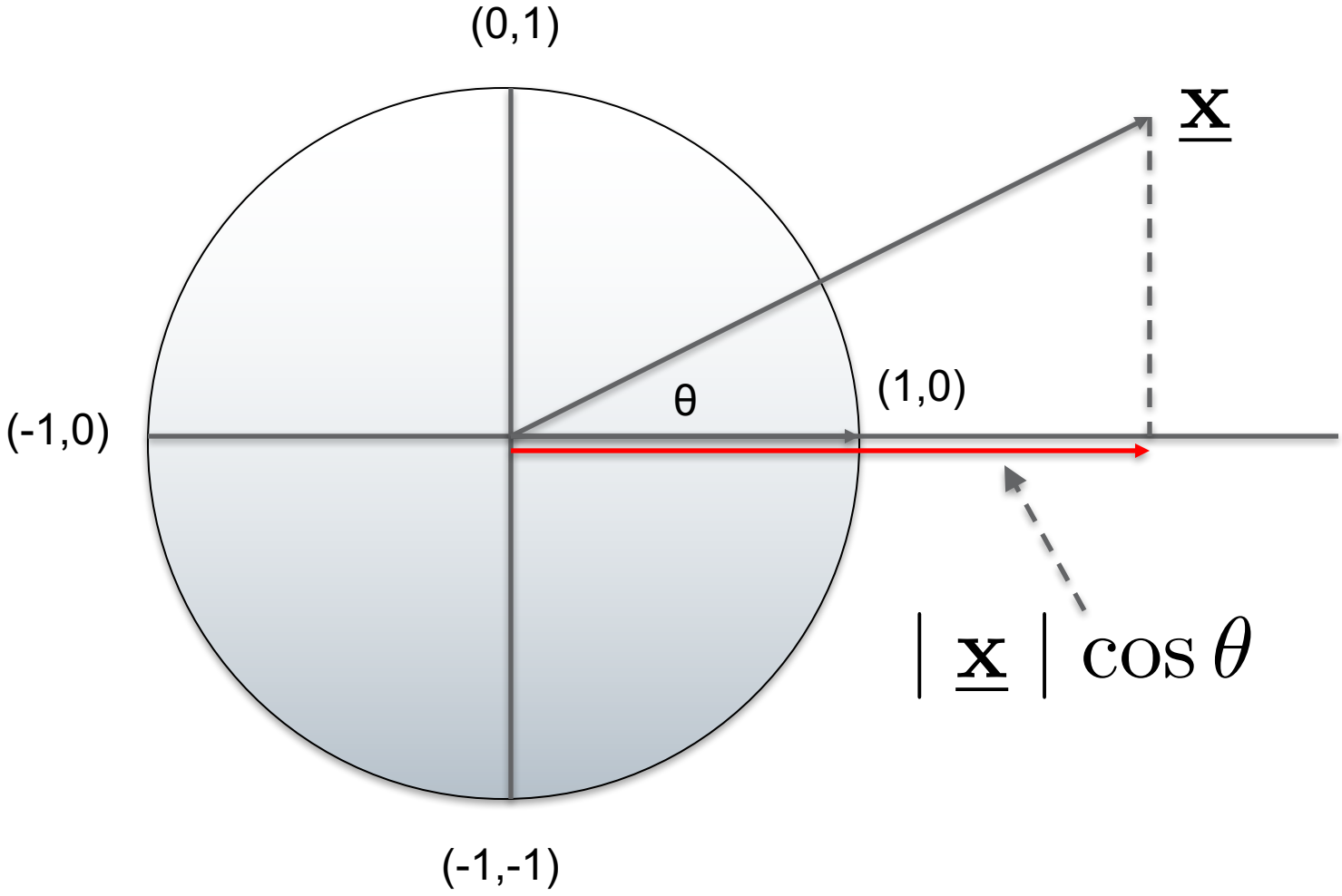
Lab 5
Transform Coding and JPEG

Image Data Compression

- **Goal: reduce the number of bits required to represent a picture**
- **How? exploit redundancies in the picture, i.e., statistical correlations**
  - Two pixels close to each other are likely to assume similar values
    - ✓ If this weren't true, pictures would look like noise
- **Technique: send only "new" information. Don't send redundant information!**
- **One standardized solution: JPEG**

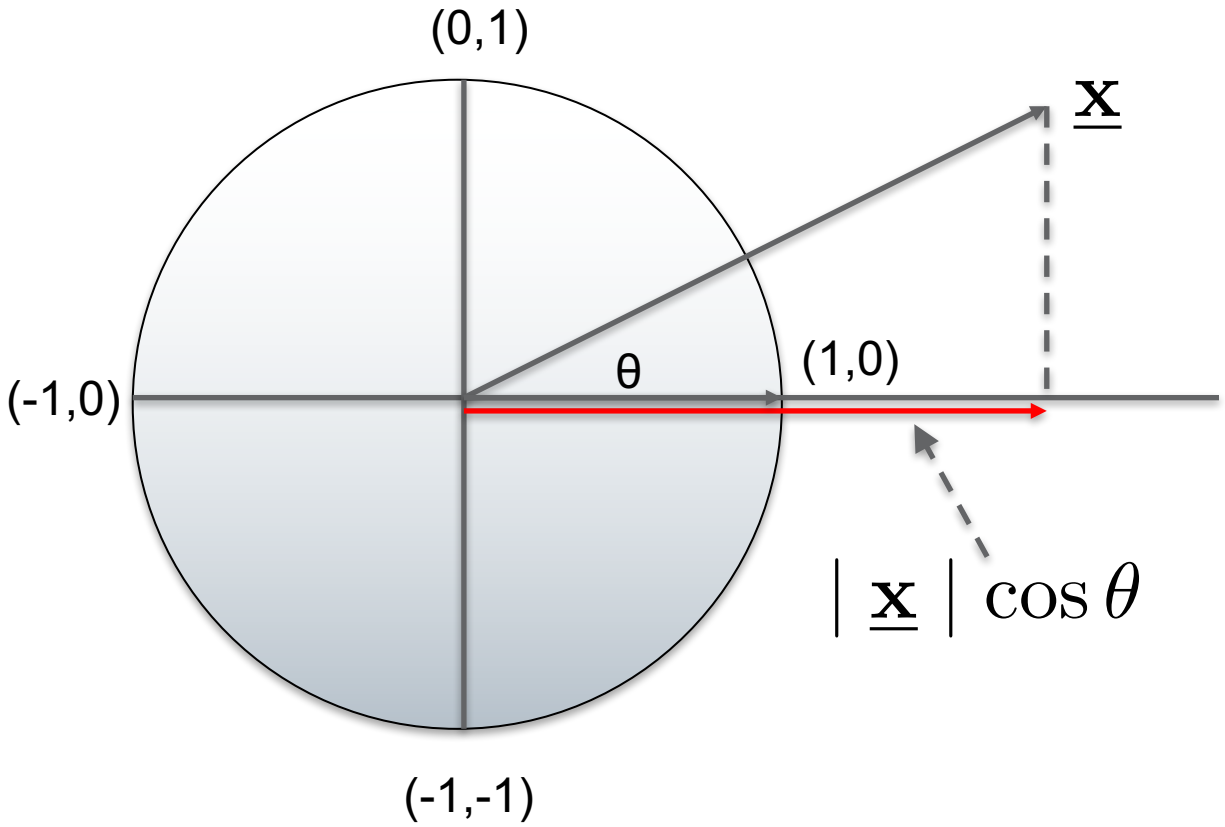
Quick Review
Projections, Transforms, Matrix Math

Projection of a vector



The Dot Product with (1,0) Gives the Projection Magnitude

$$(x_1, x_2) \cdot (1, 0) = x_1 = |\, \underline{\mathbf{x}} \,| \cos\theta$$



Important Result



The dot product of an arbitrary vector with any unit vector gives the projection of the vector in the direction of the unit vector



Matrix times a vector

- **The matrix, A, times the vector, x, is computed using the dot product**

$$\underline{\mathbf{y}} = A\underline{\mathbf{x}} = \begin{bmatrix} \underline{\mathbf{a}}_1^T \\ \underline{\mathbf{a}}_2^T \\ \vdots \\ \underline{\mathbf{a}}_N^T \end{bmatrix} \underline{\mathbf{x}} = \begin{bmatrix} \underline{\mathbf{a}}_1 \cdot \underline{\mathbf{x}} \\ \underline{\mathbf{a}}_2 \cdot \underline{\mathbf{x}} \\ \vdots \\ \underline{\mathbf{a}}_N \cdot \underline{\mathbf{x}} \end{bmatrix}$$

- **This operation defines a function: it maps x vectors to y vectors**
  - The ***x*** vector is projected onto each row of the *A* matrix

Orthonormal Basis

- **A set of vectors forms an <u>orthonormal basis</u> if**
  - Each vector is a unit vector (magnitude 1)
  - Each vector is orthogonal to every other vector

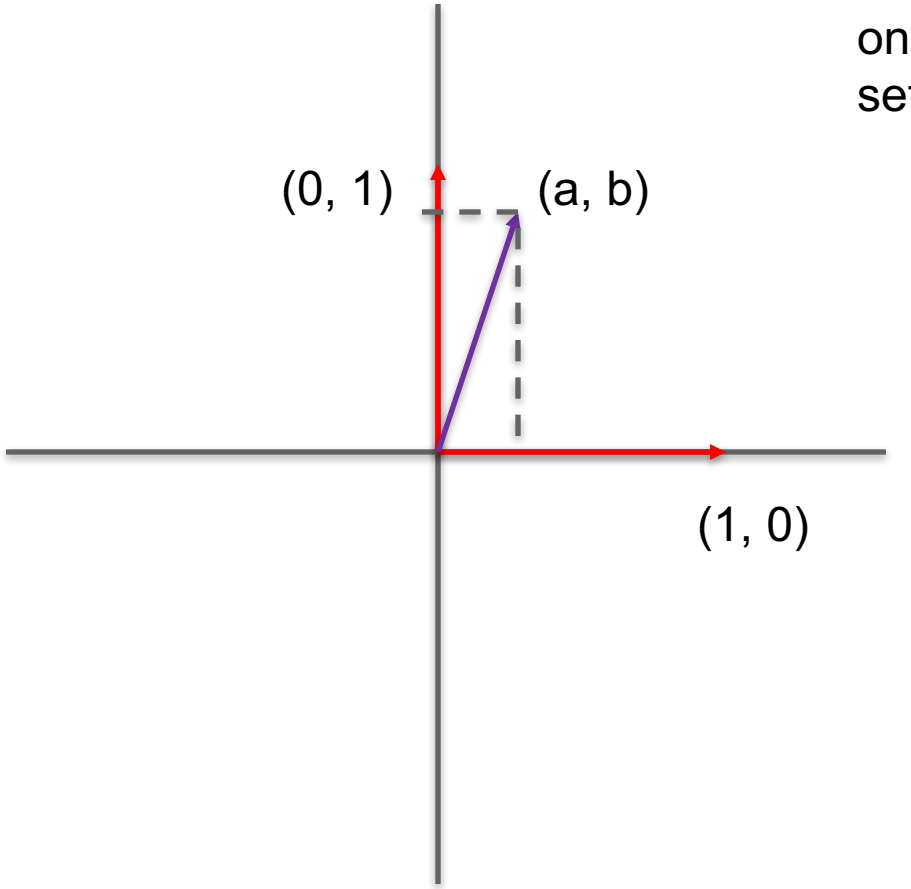
Mathematically,

$$\underline{\mathbf{u}}_i \cdot \underline{\mathbf{u}}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

Projection onto two orthonormal axes

- **The "canonical" orthonormal basis vectors in 2-D are {(1,0), (0,1)}**

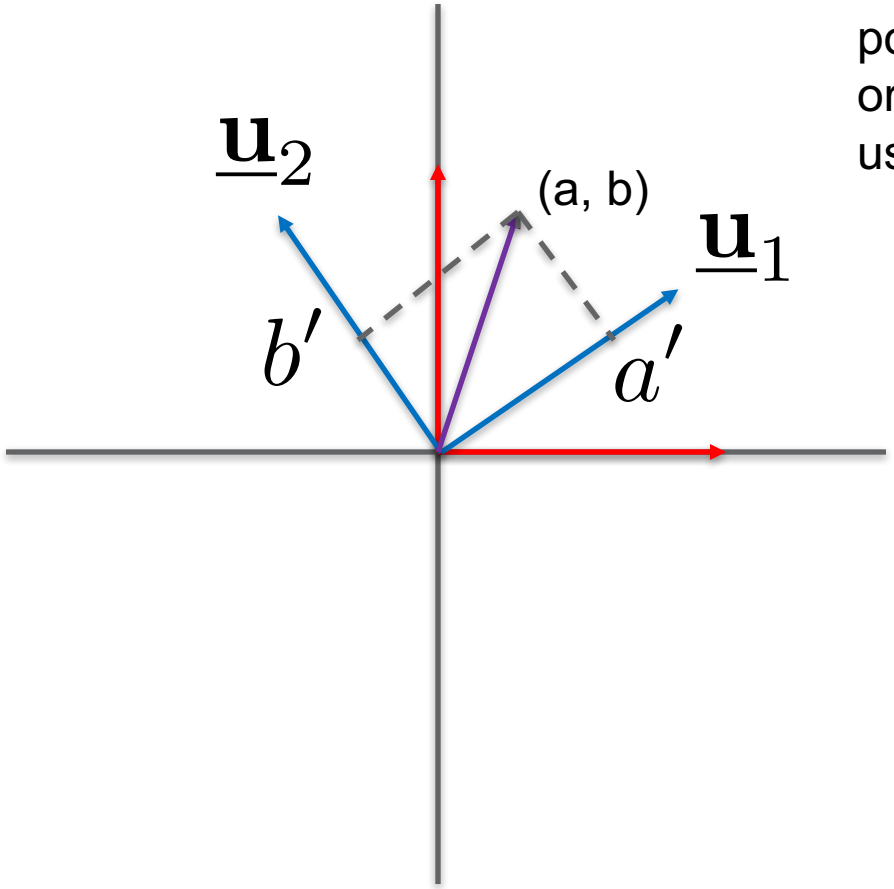
Can get the projections onto the canonical basis set with the dot product



$$(a,b)\cdot(1,0)=a$$
$$(a,b)\cdot(0,1)=b$$

Basis Vectors: A Different set

- **There are an infinite number of orthonormal basis sets!**



We can project any point onto an arbitrary orthonormal basis set using the dot product!

$$(a,b)\cdot\underline{\mathbf{u}}_1 = a'$$
$$(a,b)\cdot\underline{\mathbf{u}}_2 = b'$$

Interpretation

- **Finding the projections of a point onto a new set of unit vectors is changing the coordinate system**
  - Equivalently, it is changing the basis
- **We refer to this process as "transforming" a vector**

Matrix description of transforming a column vector

- **We can write this compactly as a matrix equation**
  - Make the basis vectors the rows of a ”transform matrix”

$$\underline{\mathbf{c}} = A\underline{\mathbf{x}}$$

$$= \begin{bmatrix} \underline{\mathbf{u}}_1^T \\ \underline{\mathbf{u}}_2^T \\ \vdots \\ \underline{\mathbf{u}}_N^T \end{bmatrix} \underline{\mathbf{x}}$$

The transform coefficients

Vector being “transformed” (i.e. for which we are computing projections onto a new basis set)

  - Every set of orthonormal basis vectors forms a “transform” via a matrix multiply

The Inverse Transform Matrix

- **We can get the original vector back from the transform coefficients as follows**

$$\underline{\mathbf{c}} = A\underline{\mathbf{x}}$$

$$A^{-1}\underline{\mathbf{c}} = \underline{\mathbf{x}}$$

- **However, because the basis vectors are orthonormal, the inverse matrix is the transpose!**

$$A^{-1} = A^{T} = \begin{bmatrix} \underline{\mathbf{u}}_1 & \underline{\mathbf{u}}_2 & \cdots & \underline{\mathbf{u}}_N \end{bmatrix}$$

Reconstructing the Original Vector from the Transform Coefficients

- **So we can easily get the original vector back from the transform coefficients**
  - The original vector is a linear combination of the basis vectors

$$\begin{aligned}\underline{\mathbf{x}} &= A^{-1}\underline{\mathbf{c}} \\ &= \begin{bmatrix}\underline{\mathbf{u}}_1 & \underline{\mathbf{u}}_2 & \cdots & \underline{\mathbf{u}}_N\end{bmatrix}\underline{\mathbf{c}} \\ &= \sum_{k=1}^{N}\underline{\mathbf{u}}_k c_k\end{aligned}$$

Matrix description of transforming a row vector

- **We can also write the transform of a row vector compactly as a matrix equation**
  - Make the basis vectors the columns of a ”transform matrix”

$$\underline{\mathbf{c}}^T = \underline{\mathbf{x}}^T \begin{bmatrix} \underline{\mathbf{u}}_1 & \underline{\mathbf{u}}_2 & \cdots & \underline{\mathbf{u}}_N \end{bmatrix}$$
$$= \underline{\mathbf{x}}^T A^T$$

The transform coefficients

Vector being “transformed” (i.e. for which we are computing projections onto a new basis set)

Parseval's Theorem

- **We can compute the energy/power in either the original domain or the transform domain**

$$\underline{\mathbf{c}} = A\underline{\mathbf{x}}$$

$$\underline{\mathbf{c}}^T = \underline{\mathbf{x}}^T A^T$$

$$\underline{\mathbf{c}}^T\underline{\mathbf{c}} = \underline{\mathbf{x}}^T A^T A\underline{\mathbf{x}}$$

$$\underline{\mathbf{c}}^T\underline{\mathbf{c}} = \underline{\mathbf{x}}^T I\underline{\mathbf{x}}$$

$$\sum_{k=1}^{N} c_k^2 = \sum_{n=1}^{N} x_n^2$$

The DCT

The 1-D Discrete Cosine Transform (DCT)

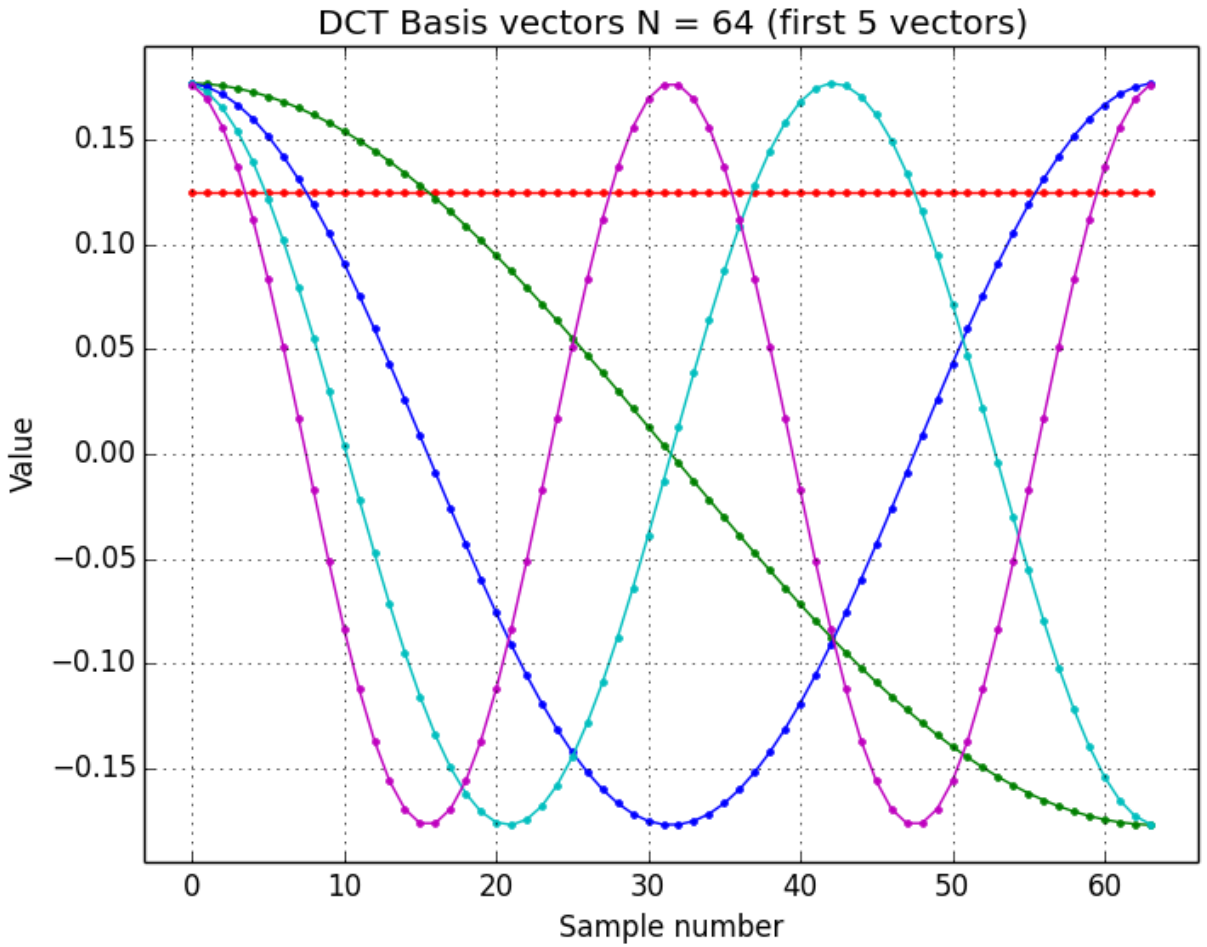
- **The 1-D DCT is a matrix multiply as you would expect**
  - The basis vectors for the DCT are the following sinusoids

$$\underline{\mathbf{u}}_{k+1} = \alpha(k)\cos\left(2\pi\left(\frac{k}{2N}\right)n + \frac{\pi}{2N}k\right)$$

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{2}{N}} & \text{for } k = 1, 2, \ldots, N-1 \end{cases}$$

The DCT Basis Functions

- **The 1-D DCT is a matrix multiply as you would expect**



The 2-D Discrete Cosine Transform

- **Start with a 2-D array *X***
- **First 1-D transform all the columns to get an intermediate matrix**
- **Then transform all the rows of the intermediate matrix to get the final matrix**
- **This can be written concisely as the product of 3 matrices**

$$C = \underbrace{AX}\, A^T$$

xforms the columns of X

Multiplying *AX* by A^T xforms the rows of *AX*

Reconstruction

- **Recall that**

$$A^{-1} = A^T$$

- **We can therefore recover *X* from *C* via**

$$X = A^T C A$$

- **Equivalently**

These are basis images

$$X = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \underbrace{\mathbf{u}_i \mathbf{u}_j^T}$$

Superposition of basis images

- **The array *X* is reconstructed as a superposition of basis images**

2-D basis images

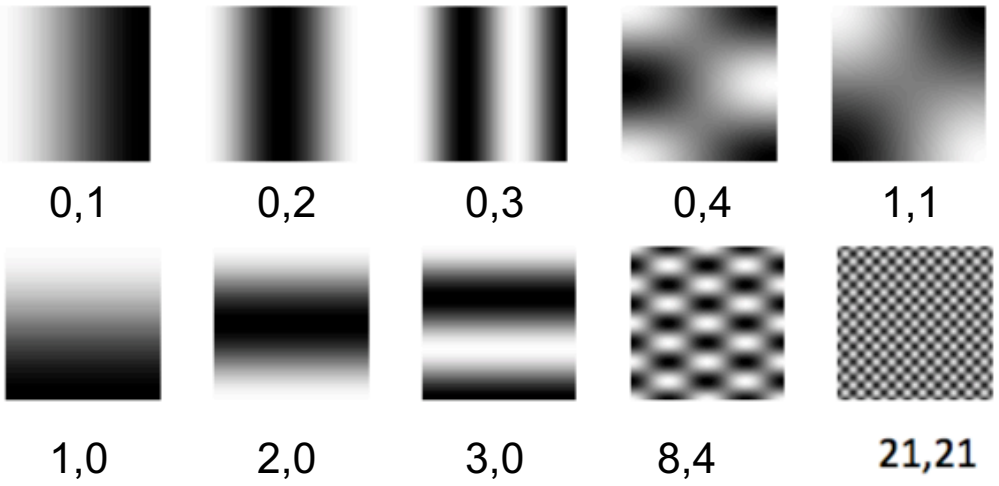


Figure 1: Some basis images where "i, j" means $\mathbf{u}_i\mathbf{u}_j^T$ for $N = 64$

Transform coding and JPEG

Image partitioning into 8-by-8 blocks is common

CU Boulder

Original image

Keep only DC coefficient of each block

Keep 2x2 coefficient square

Keep 3x3 coefficient square

CU Boulder

JPEG History

- **Work began in 1986. Standard finalized in 1992**
- **Widely used for photography, the internet, and image exchange**
- **Motion JPEG common for video editing**
  - Each frame coded individually allows frame-by-frame access
- **Core technologies**
  - 8-by-8 DCT transform
  - Uniform quantization with varying step sizes per transform coefficient
  - DPCM coding of DC coefficient
  - Zig-zag scan, run-length encoding, huffman coding