University of Stavanger

Prof. Kjersti Engan

# ELE510 Image processing and computer vision

Image Transformations ( SVD ), 2020

# Introduction

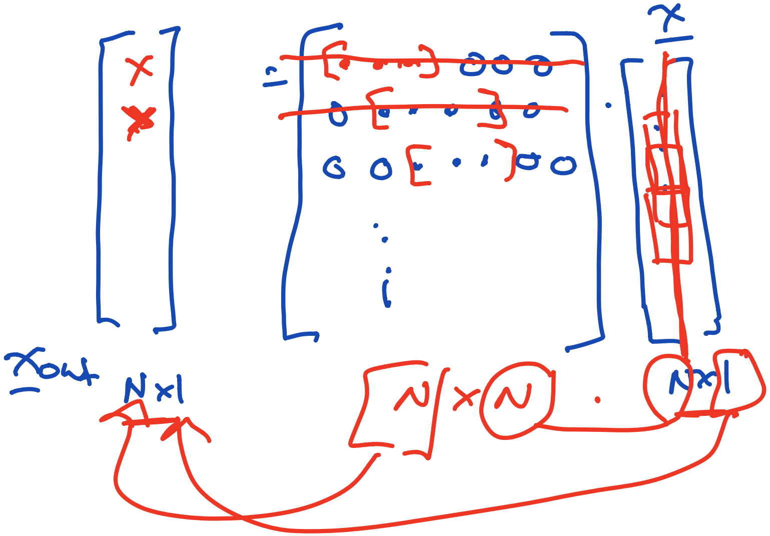
- Often we want to find **operators** that **transform** an image to a more «efficient» form. We are looking for a **linear superposition** *of* ***elementary images***.
- In order to achieve this goal we will use the **separable operator.**
- We represent the images and the left and right operator as 2D square matrices of *size (NxN)*. $\underbrace{\text{F is input image,}}_{N \times N}$ $\underbrace{\text{G is output image.}}_{N \times N}$

$$\boxed{\mathbf{G} = \mathcal{H}_c^T \mathbf{F} \mathcal{H}_r}$$

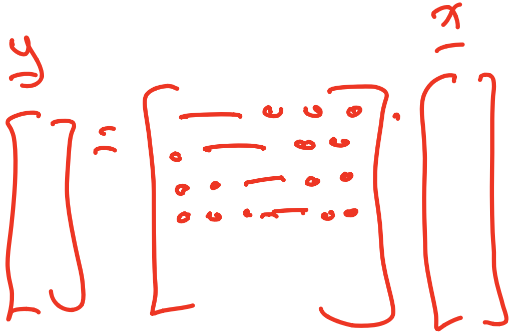
$\mathcal{H}_c^T$ from the left, works on image columns

$\mathcal{H}_r$ from the right, works on image rows

# Matrix – vector look at filtering



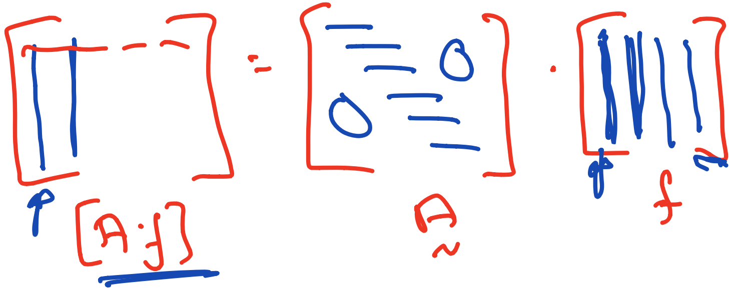

kernel $k = [\,\cdot\;\cdot\;\cdot\;\cdot\,]$



# Seperable filters in 2D, matrix-vector formulation

Separable filter: filter over the column first, and tehreafter filter the results over the rows.
This corresponds to 2 x 1D filtering instead of 2D filter

$f$ : NxN input image    $g$ : NxN output image




$$\tilde{H}_c^T \tilde{f} \tilde{H}_r$$

position of transpose depends on how the matrix is defined.

filtering over the columns

$$g^T = \tilde{B} \cdot [Af]^T$$

$$g = [B \cdot [Af]^T]^T = [Af]^{TT} \cdot B^T$$

$$\boxed{g} = [A \cdot f] \cdot B^T = \boxed{A \cdot f \cdot B^T}$$

filtering over the rows.

# The inverse operator

$$\mathbf{G} = \mathcal{H}_c^T \mathbf{F} \mathcal{H}_r$$

F : input image

$$\mathcal{H}_c^{-T} \mathbf{G} \mathcal{H}_r^{-1} = \overbrace{\mathcal{H}_c^{-T} \mathcal{H}_c^T}^{=I} \mathbf{F} \underbrace{\mathcal{H}_r \mathcal{H}_r^{-1}}_{I} = \mathbf{F}$$

$$\mathbf{F} = \mathcal{H}_c^{-T} \mathbf{G} \mathcal{H}_r^{-1}$$

We now partition the inverse operators in column and row vectors, respectively:

$$\mathcal{H}_c^{-T} \equiv [\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_N], \quad \mathcal{H}_r^{-1} \equiv \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_N^T \end{bmatrix}$$

$v_i$ :

# Expansion of image F in terms of vector outer products

---

$$\boxed{\mathbf{F}} = \underbrace{[\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_N]}^{H_c^{-T}}\mathbf{G}\underbrace{\begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_N^T \end{bmatrix}}_{H_r^{-T}} = \boxed{\sum_{i=1}^{N}\sum_{j=1}^{N} g_{ij}\boxed{\mathbf{u}_i\mathbf{v}_j^T}}$$

$$\tilde{G} = \begin{bmatrix} g_{11} & g_{12} & g_{13} & \cdots \\ g_{21} & & & \\ g_{31} & & \ddots & \\ \vdots & & & \end{bmatrix}$$

The image **F** is now expressed as a sum of ***elementary images*** given by the outer product of the vectors **u** and **v**. The coefficient gij is multiplied by the elementary image **F**ij expressed as:

$$\boxed{\mathbf{F}_{ij} = \mathbf{u}_i\mathbf{v}_j^T}$$

# How do we choose the transforming matrices?

- So that the transformed image can be represented by fewer bits, i.e. compression.
- Smoothing by omitting high frequency components.
- Approximating the input image according to some defined criteria.

It is convenient to choose a transformation that is easily inverted!

This can be achieved by a **unitary transform**, the transformation matrices are unitary. The result, **G**, is called the unitary transform domain of image **F**.

# Unitary Transforms

First we define a ***unitary*** matrix:

$$\mathbf{U}\mathbf{U}^{T*} = \mathbf{U}\mathbf{U}^{H} = \mathbf{I} \quad \text{where} \quad \mathbf{I} \text{ is the unit matrix}$$

*H* , the ***Hermitian*** is the same as the ***conjugate transpose***. If the elements of the matrix U are real numbers we use the term ***orthogonal*** instead of unitary.

The inverse of a unitary matrix is the complex conjugate of its transpose.

$U^{-1} = U^H$

$UU^{-1} = I$

$UU^H = I$

Replace the operator matrices with the unitary matrices **U** and **V** and we get:

$$\mathbf{F} = \mathbf{U}^{*}\mathbf{G}\mathbf{V}^{H} \quad \text{or for real matrices} \quad \mathbf{F} = \mathbf{U}\mathbf{G}\mathbf{V}^{T}$$

# Singular Value Decomposition (SVD)

$$\underbrace{F}_{N \times N} = U \underbrace{G}_{N \times N} V^T$$

Background: If we can construct a matrix **G** that is ***diagonal*** with unitary matrices **U** and **V**, the image **F** is written as a sum of N elementary images. Diagonalization is in general only possible for square matrixes. If a matrix is square and symmetric, then we can always diagonalize it.

In order to diagonalize an image we construct the new square and symmetric image:

$\mathbf{F}\mathbf{F}^T$ , assume that this matrix is of rank $r$ then

$\mathbf{F} = \mathbf{U}\Lambda^{\frac{1}{2}}\mathbf{V}^T$ , where

$\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices of size $(N \times r)$ and

$\Lambda^{\frac{1}{2}}$ is a diagonal $(r \times r)$ matrix

$$F = \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{N} g_{ij} \, \underline{u}_i \underline{v}_j^T}_{N^2}$$

$$G = \begin{bmatrix} g_{11} & & & 0 \\ & g_{22} & & \\ & & \ddots & \\ 0 & & & \ddots \end{bmatrix}$$

$$F = \underbrace{\sum_{i=1}^{N}}_{N} g_{ii} \, \underline{\underline{u}_i \underline{v}_i^T}$$

## How do we compute the eigenvector matrices **U** and **V** and the *eigenvalues* ?

$$\mathbf{F}\mathbf{F}^T = \mathbf{U}\Lambda\mathbf{U}^T \quad \text{and} \quad \mathbf{F}^T\mathbf{F} = \mathbf{V}\Lambda\mathbf{V}^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{bmatrix}$$

If the number of non-zero eigenvalues is *r*, then we can write:

$$\mathbf{F} = \sum_{i=1}^{r} \lambda_i^{\frac{1}{2}} \mathbf{u}_i \mathbf{v}_i^T \quad \text{where} \quad \mathbf{u}_i \mathbf{v}_i^T \; i \in \{1, 2, \cdots, r\} \text{ are the eigenimages}$$

***Image approximation*** by keeping k<r terms:

$$\boxed{\mathbf{F}_k = \sum_{i=1}^{k} \lambda_i^{\frac{1}{2}} \mathbf{u}_i \mathbf{v}_i^T}$$

$$\mathbf{F}: \underset{m \times n}{\begin{bmatrix} & \end{bmatrix}} = \underset{m \times m}{U} \cdot \underset{m \times n}{\Lambda^{\frac{1}{2}}} \, \underset{n \times n}{V^T}$$

the rank $r \leq \min(m, n)$

# Note on the use of SVD to represent a given arbitrary image **F**

- The eigenimages are specific for each input image **F**. Different images have different eigenimages.
- The eigenvector matrices must be computed separately for each given image.
- If we arrange the eigenvalues in decreasing order and truncate the expansion at some integer *k<r* we get the least square approximation of the image **F**, with error given by:

Fii

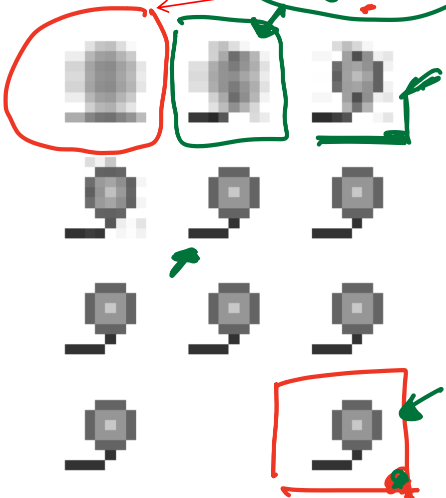
$$\mathbf{D} = \mathbf{F} - \mathbf{F}_k = \sum_{i=k+1}^{r} \lambda_i^{\frac{1}{2}} \mathbf{u}_i \mathbf{v}_i^T \quad \text{where} \quad \|\mathbf{D}\| = \text{trace}\left(\mathbf{D}^T \mathbf{D}\right) = \sum_{i=k+1}^{r} \lambda_i$$

The SVD is optimal in the ***least square error sense***, but the basis images (eigenimages) are determined by the image itself.

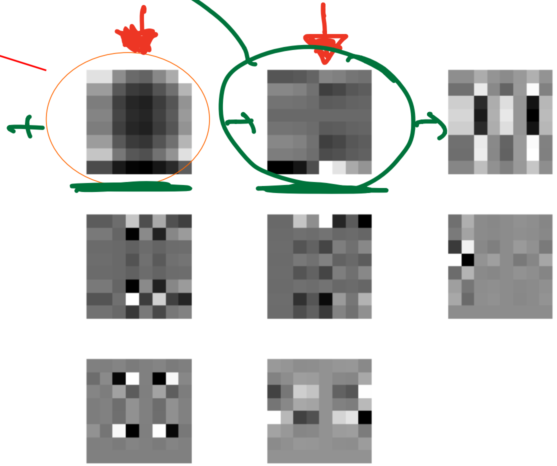
# SVD – example 1

$g_2 \cdot u_2 v_2^T$ +

$g_1 \cdot \mathbf{u}_1\mathbf{v}_1^T$



8x8

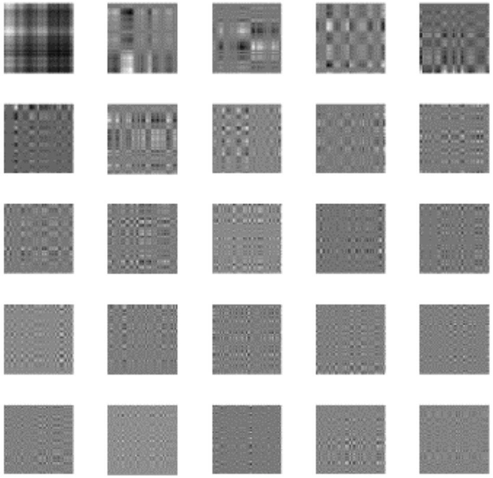


Eigenimages

Representations using 1, 2, 3 … 10 eigenvalues.
Original 8x8 image **F** lower right.

5 coefficients instead of 64 pixels

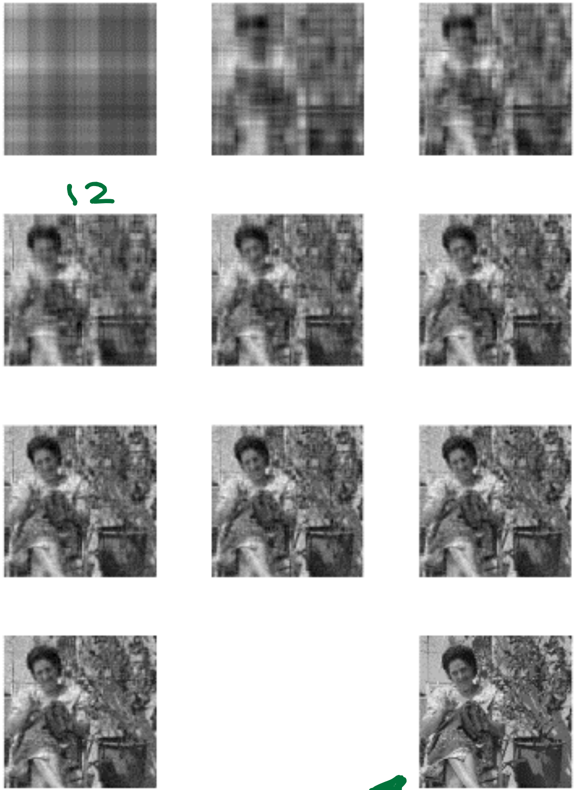
# SVD – example 2



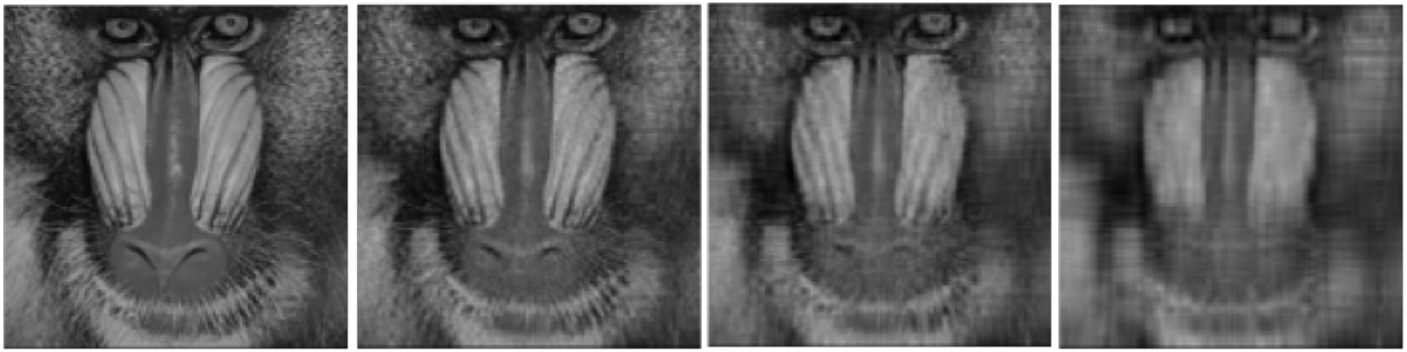
First 25 ***eigenimages***, left.

***Representation*** using 1, 2k, 3k, … 10k eigenvalues, with k = 3, right.

***Original image***: lower right, size 512x512.



# SVD – example 3



Original and three SVD representations using 32, 16, and 8 basis vectors

# Comments

Are there any set of elementary images in terms of which any image may be expanded?

Yes. They are defined in terms of ***complete*** and ***orthonormal*** sets of discrete valued discrete functions

Examples are the set of

- Haar functions
- Walsh functions
- Hadamard
- Discrete wavelet transform

We will in the following discuss the ***Discrete Fourier Transform***