Unitary Transforms

Computer Vision

Image Decompositions

Today's Overview:

- Scale-space
- Unitary transform:
    - What are they
    - How to define bases / decomposition
    - Properties
    - Sample transforms
- PCA: Domain-specific transforms

Computer Vision

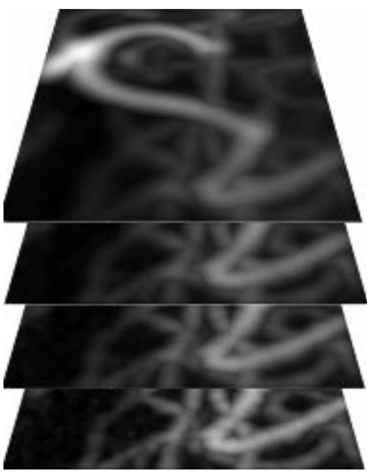
Scale Space

Computer Vision

Scale space: motivation

One way to decompose images, since scenes contain information at different levels of detail.

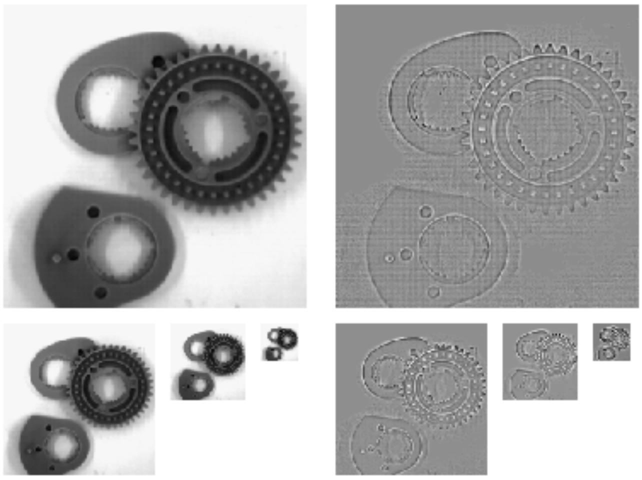
Psychophysical and neurophysiological relevance



1. Increases efficiency by sometimes working on lower resolutions
2. Helps develop hierarchical descriptions

Computer Vision

Scale space: Gaussian-Laplacian pyramid



For image I_i

1. Smooth I_i (with Gaussian) => S_i
2. Take difference image:
(since DoG ~= Laplacian)
$$L_i = I_i - S_i$$
3. Reduce smoothed image size:
$$I_{i+1} = \text{down-sample}(S_i)$$

The 3rd step is allowed following the Nyquist theorem (i.e., given sufficient smoothing).

Zero-crossings of the Laplacian yield edges, thus interesting information in the Laplacian pyramid; e.g., important edges coincide spatially at all scales

Computer Vision

Scale space: in discrete domain

Discrete approximations of the Gaussian filters should ensure not to generate spurious structures!

e.g. for a small (3x1) smoothing filter
with positive coefficients c_{-1}, c_0, c_1

$$\Downarrow$$

make sure that $c_0^2 \geq 4c_{-1}c_1$

i.e., [1,2,1] is a valid scale space filter,
whereas [1,1,1] is not.

Computer Vision

Unitary Transforms

Computer Vision

Motivation example

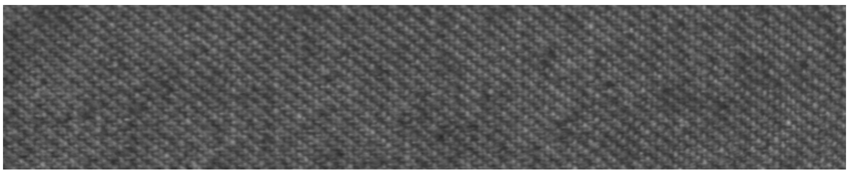


Image in pixel-space

Task: how to find thickness of repeating pattern

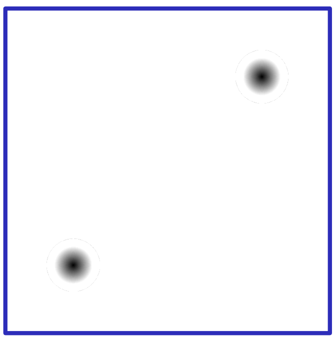


Image in Fourier space

Instead of counting peaks, etc;
we can find the maximum in DFT,
and take corresponding spatial size

It is still the (same) "image",
with no *more* or *less* info.

But, more useful in this
domain for our purpose

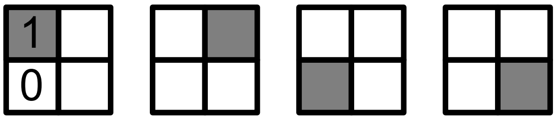
Unitary image transforms

Image decomposition into a family of
orthonormal *basis images*

Decomposition as linear combination of basis vectors/images

Examples so far:

Pixelwise decomposition: 1 Dirac impulse at the corresponding pixel in each basis image
(perfect localization in image space, none in frequency)



Example: For 2x2 images

Fourier decomposition: 1 oriented cosine/sine pattern in each basis image
(perfect localization in frequency domain, none in space)

Unitary operators

Unitary operator U is a matrix of all bases as row vectors

They preserve the inner product, i.e. $U^* U = U U^* = I$
Or, equivalently $U^{-1} = U^*$

For real funcs, only possible (iff) columns of U are orthonormal
(orthonormal: inner-product of all components with self =1, others =0)

- Pixelwise/Fourier have orthonormal basis images

- Fourier transform (follows from Parseval's theorem)

- Rotations are unitary (does not change vector lengths)

Computer Vision

Unitary transforms

Properties:

- Concentrate energy in a few components, i.e. only few basis images that can faithfully represent
- Compromise localization in space/frequency

(other examples of decompositions given later for more balanced localizations in different spaces)

Image independent rotations

(rotations, because new axes are also orthonormal + Euclidean distance preserved)

(image independent transforms are generic but suboptimal, as opposed to PCA that we will see later)

E.g.: decomposition as Dirac impulses or Fourier domain is decided without knowing type/content of images

Computer Vision

Basis images: Orthonormal

Orthonormal basis images B conform:

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} B_i(x,y)\;\; B_j^*(x,y) = \delta_{ij}$$

necessary and sufficient

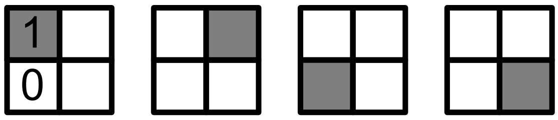
with * indicating the complex conjugate, because

- We do want basis images linearly independent of each other ➔ orthogonal: Bi Bj=0
- We do not want an all zero basis B, which would generate zero under any linear combination, thus be useless in representing anything
- Not to change dimension scales, better to have a unit length B ➔ thus Bi Bi=1

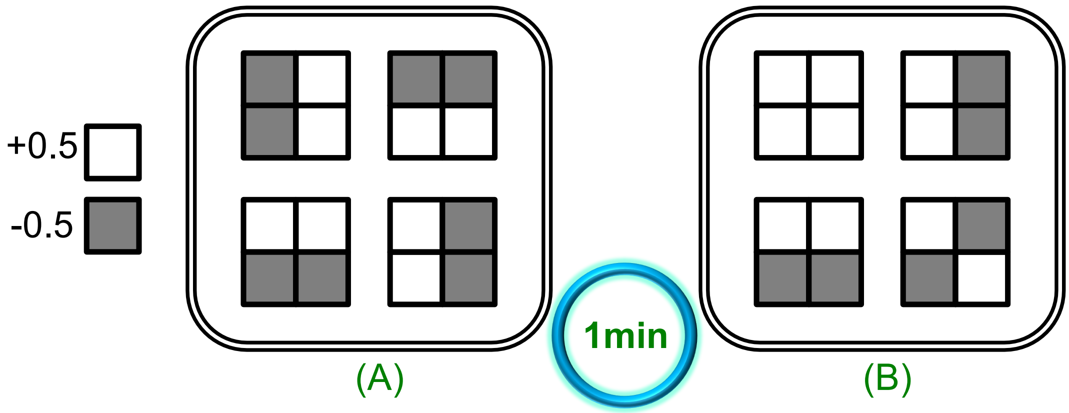
Basis images: Orthonormal

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} B_i(x,y)\ \ B_j^*(x,y) = \delta_{ij}$$

Let's see if this holds for Dirac impulses in pixelwise decomposition:



Can these be unitary decompositions?



Orthogonality of functions (e.g. trigonometric)

Example: period $P = \dfrac{2\pi}{\omega}$ of $\cos m\omega\, x$ for $m = 1,2,\ldots$

$$\int_0^p \cos m\omega\, x\ \cos n\omega\, x\ dx\ = \delta_{mn}\frac{P}{2}$$

$$\int_0^p \cos m\omega\, x\ \sin n\omega\, x\ dx\ = 0$$

$$\int_0^p \sin m\omega\, x\ \sin n\omega\, x\ dx\ = \delta_{mn}\frac{P}{2}$$

For all positive values of m=1,2,… a countable set of orthogonal functions is generated

Generalization of orthonormality to vector calculus towards infinite dimensions (Hilbert spaces)

Computer Vision

Orthogonality of functions (e.g. trigonometric)

Example: period $P = \frac{2\pi}{\omega}$ of $\cos m\omega x$ for $m = 1,2,...$

$$\int_0^p \cos m\omega x \; \cos n\omega x \; dx \; = \delta_{mn} \frac{P}{2}$$

$$\int_0^p \cos m\omega x \; \sin n\omega x \; dx \; = 0$$

$$\int_0^p \sin m\omega x \; \sin n\omega x \; dx \; = \delta_{mn} \frac{P}{2}$$

Problems with infinite dimensions: representation need not be unique (e.g. aliased freqs) & may not be complete (even funcs)

These problems disappear with discretization

Computer Vision

Completeness condition

Arbitrary square-integrable functions can be characterized by their correlations with the basis set of orthonormal functions

To represent (N x 1) sample vectors, any N orthogonal bases will be complete!

But, how can we find these bases?

In the discrete case, the problem is how to find **sufficient** number of orthogonal basis functions.

Computer Vision

Completeness condition

An example with 16 samples:

- Cos set is all **orthogonal, BUT** they repeat (9 & 7 are identical)
- To no surprise, **odd funcs** cannot be represented by cos set
- Sine can represent odds, thus Fourier basis funcs is a complete set
- This yields **16 orthogonal complex** trigonometric basis funcs

E.g. $\cos\frac{2\pi}{16}ux$ x = 0,1,2…15, and u = 0,1,…,8

other u's identical, but signs reversed; e.g. u=7 & u=9 identical

$\sin\frac{2\pi}{16}ux$ u = 1…7 functions with $u=0$ and $u=8$ vanish

Hence, 16 Fourier basis funcs of form: $\frac{1}{N}e^{-2\pi i\frac{ux}{N}}$

Computer Vision

Basis images: Separable

Not a requirement, but preferred

1-D → higher dimensions

$$B_{ij}(x,y) = \phi_i(x)\ \psi_j(y)$$

Or, equivalently

$$B_{ij} = \phi_i \psi_j^t$$

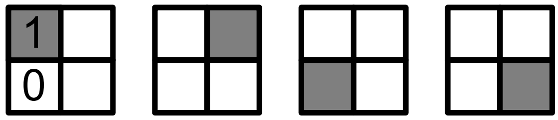
(can be decomposed into products of 1D functions)

With *separable* basis images, many image analysis operation can be run faster (small kernel, separately in each axis)

Pixelwise (Dirac) is separable, i.e. abscissa and ordinate,
But many basis functions are not separable.

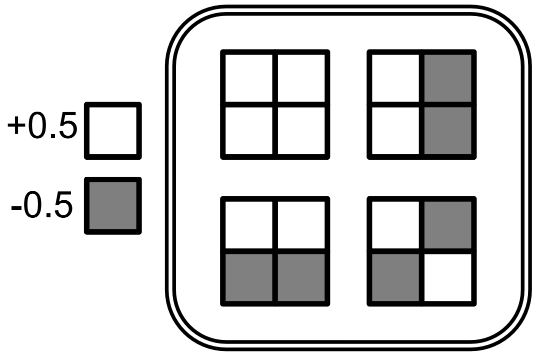
Orthonormality

Pixelwise:



φ_i^t :   [1 0], [0 1]

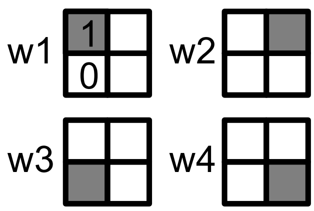
Is this (unitary) decomposition separable?
If so, what are φ_i^t?



Decomposition of images

Now we decided the bases B, but how to find the representation of a given image, i.e. basis weights w_{uv}

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv} B_{uv}(x,y)$$



For a given basis $B_{u'v'}$ in order to find the weight $w_{u'v'}$

Multiply with bases

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) B^*_{u'v'}(x,y)$$

Use def above

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv} B_{uv}(x,y)\right) B^*_{u'v'}(x,y)$$

Shift x,y inside

$$= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv} \left(\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} B_{uv}(x,y) B^*_{u'v'}(x,y)\right)$$

Given orthonorm.

$$= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv} \delta_{u'v'}$$

$$= w_{u'v'}$$

Computer Vision

Decomposition of images: Summary

$$\boxed{f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv} B_{uv}(x,y)}$$

cf. projection of vector onto basis vectors or interpret as correlation with reference patterns

Transformed image: $F(u,v) = w_{uv}$

Forward transform:

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)B_{uv}^{*}(x,y)$$

Backward transform:

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)B_{uv}(x,y)$$

Computer Vision

Optimal truncation property

If only a small number of bases are to be retained, optimally **which weights should be used for those**

Optimal truncation property states that these weights should be the same as original ones!

WHY?? Not so obvious as it seems...

(intuitive description as "any other combination not being able to represent/explain the *missing* info from missing bases due to orthogonality of the bases")

Computer Vision

Optimal truncation property

For a formal proof, when M'N' dimensions are kept
set a GOAL to find the truncated decomposition

$$\hat{f}(x,y) = \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} c_{uv} B_{uv}(x,y)$$

with M' < M and N' < N, such that
it minimizes the approximation error (i.e. closest fit)

$$e_{M'N'} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(f(x,y) - \hat{f}(x,y)\right)^2$$

w_{uv} that minimize $e_{M'N'}$ are given by:

$$w_{uv} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) B^*_{uv}(x,y)$$

Show that these weights are indeed the ones from the original decomposition

Computer Vision

Optimal truncation property

Proof : Show that other weights c_{uv} → larger $e_{M'N'}$

$$e_{M'N'} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(f(x,y) - \hat{f}(x,y)\right)^2 \qquad \boxed{c_{uv} = w_{uv} - (w_{uv} - c_{uv})}$$

Definition

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left| f(x,y) - \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} c_{uv} B_{uv}(x,y)\right|^2$$

Reparametrize

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left| f(x,y) - \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} w_{uv} B_{uv}(x,y) + \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} (w_{uv} - c_{uv}) B_{uv}(x,y)\right|^2$$

Remaining terms from f summation & integrate over x,y

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left| \sum_{u=M'}^{M-1}\sum_{v=N'}^{N-1} w_{uv} B_{uv}(x,y)\right|^2 + \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} \left|w_{uv} - c_{uv}\right|^2$$

Last term is positive and is minimized for $c_{uv} = w_{uv}$

Computer Vision

Optimal truncation property

This theorem underlies the use of unitary transforms for *image compression* applications:

Energy in images tends to be concentrated in lower frequencies
Thus taking more terms (where $c_{uv} = w_{uv}$) always improves the approximation, i.e.

$$e_{M'N'}$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left| \sum_{u=M'}^{M-1} \sum_{v=N'}^{N-1} w_{uv} B_{uv}(x,y) \right|^2$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(\sum_{u=M'}^{M-1} \sum_{v=N'}^{N-1} \left| w_{uv} \right|^2 \right)$$

Computer Vision

Examples of unitary transforms

Assuming square images

- 1. Cosine transform
- 2. Sine transform
- 3. Hadamard transform
- 4. Haar transform
- 5. Slant transform

Generally, we seek decompositions with strong compaction; driven by practical experience and implementation efficiency

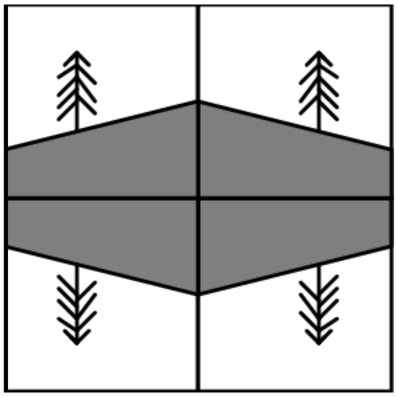
Cosine transform gives best decorrelation

Computer Vision

Cosine transform

Converts Fourier transform into a real transform and helps suppress spurious high frequencies.

We extend the image around a corner:



The extended image is even!

So, only even funcs (cosines) can represent it, and the image now wraps around continuously

Computer Vision

Cosine transform

DFT of the extended image:

$$F_e(u,v) = \frac{1}{4N^2} \sum_{x=-N}^{N-1} \sum_{y=-N}^{N-1} f_e(x,y) e^{-2\pi i \left(\frac{u(x+1/2)}{2N} + \frac{v(y+1/2)}{2N} \right)}$$

Domain [-N .. N], normalized by $4N^2$

Because $f_e(x,y)$ is even, sines disappear, thus:

$$\frac{1}{N^2} \sum_{x=-N}^{N-1} \sum_{y=-N}^{N-1} f_e(x,y) \cos\left(\frac{\pi}{N} u(x+1/2)\right) \cos\left(\frac{\pi}{N} v(y+1/2)\right)$$

Real-valued,

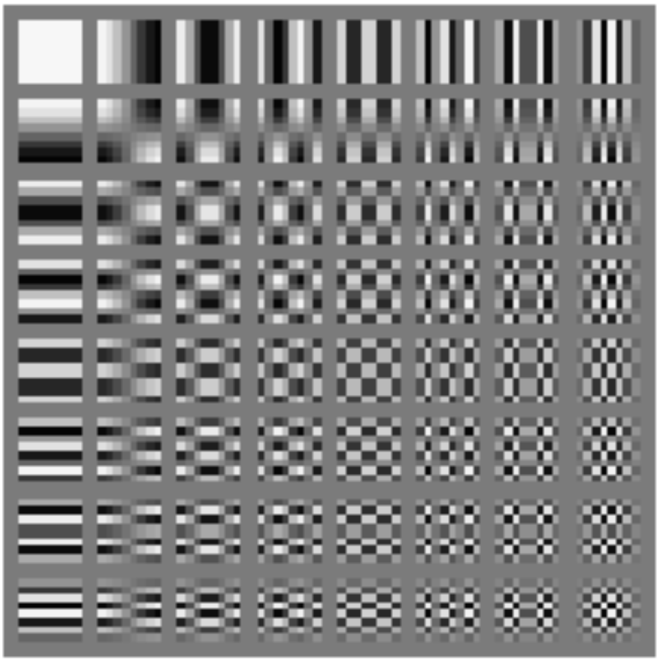
Even to represent periodic image space,

Also separable

Computer Vision

Discrete Cosine Transform (DCT)

8 x 8 basis images:



Computer Vision

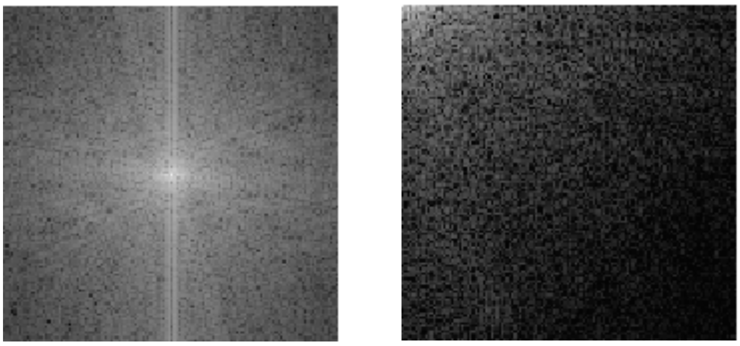
Discrete Cosine Transform (DCT)

1. Eliminates the boundary discontinuities
2. Components are well decorrelated
3. Requires real computations only
4. Has fast $\mathcal{O}(n \log n)$ implementations
5. DCT chips are available
6. Was long time the most popular compression basis

Computer Vision

DFT vs. DCT

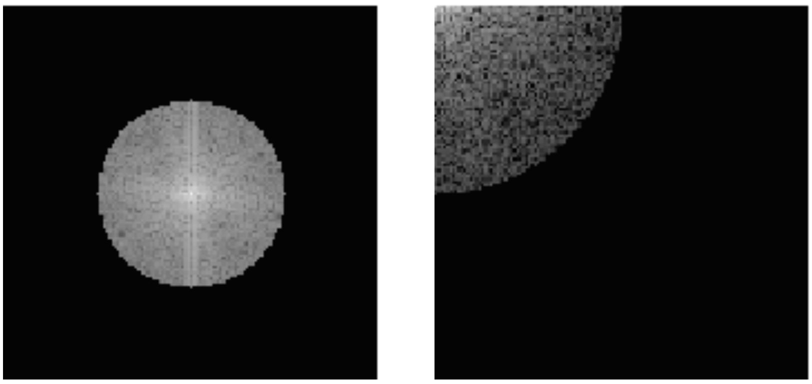
Left : DFT, right : DCT



Computer Vision

DFT vs. DCT

Zonal truncations:



When the same number of samples are retained in both cases (i.e., same compression ratio)

Computer Vision

DFT vs. DCT

DFT

Horizontal top/bottom ripple, spurious high frequencies

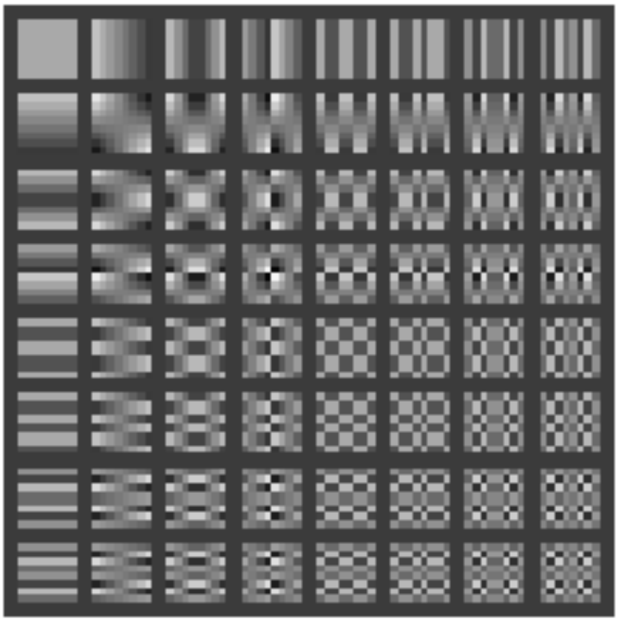
DCT

Computer Vision

Slant transform

on the basis of slant matrices

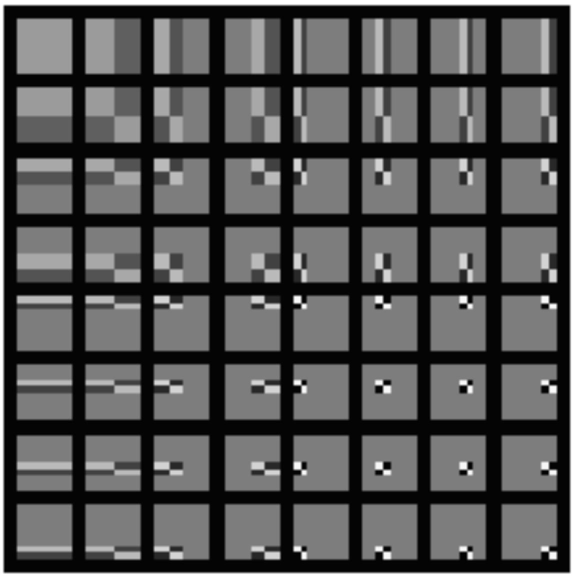
e.g. basis images for 8 x 8 :



discrete sawtooth-like basis vectors which efficiently represent linear brightness variations along an image line

Computer Vision

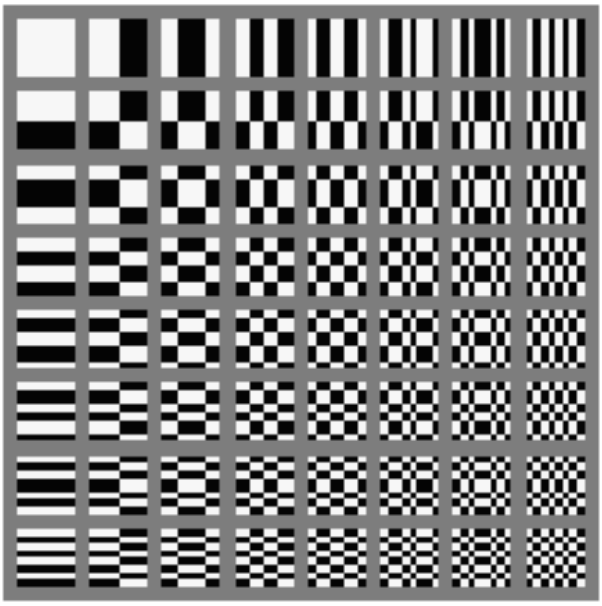
Haar transform



- Is an example of a wavelet transform
- localised both in space and in terms of frequency
- Note also that for higher frequencies, the spatial extent gets smaller, a typical feature of wavelets

Computer Vision

The Hadamard transform



- Only 1s and -1s, therefore no multiplication needed: one of the first for HW implementation
- Recursive operation of [1 1; 1 -1]
- Generates minimally correlated binary blocks
- Binary ➔ efficient ➔ barcode reading
- All examples had same orthogonal set for rows&cols, BUT need not be so, e.g. Haar X Hadamard possible

Computer Vision

Principal Component Analysis

Computer Vision

Principal component analysis: Motivation

Image independent transforms are suboptimal

PCA, a.k.a. Karhunen-Loève Transform (KLT)

extracts statistics from images for a **customized** orthogonal basis set with uncorrelated weights

PCA: technique based on eigenvectors of the covariance matrix

Computer Vision

PCA

<u>Central idea:</u>

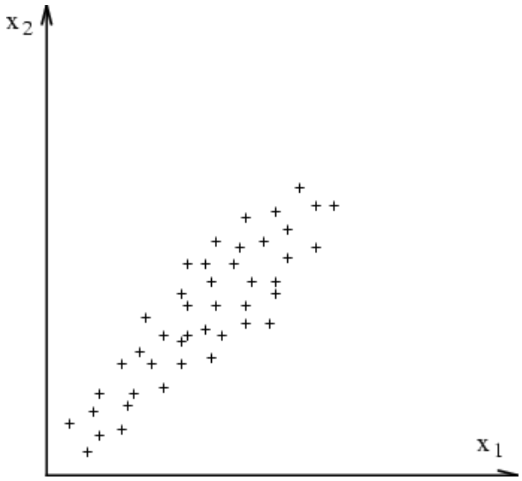
Reduce the dimensionality of data consisting of many interrelated variables, while retaining as much as possible of the variation

Achieved by transforming to new, uncorrelated variables, the principal components, which are ordered so that the first few retain most of the variation

Remarks:
- Receiver needs the bases (contrary to generic transforms)
- For a diverse input set, PCA will resemble DCT (DCT is the generic transform with optimal decorrelation)
- Use cases: feature selection, classification or inspection

Computer Vision

Correlated variables

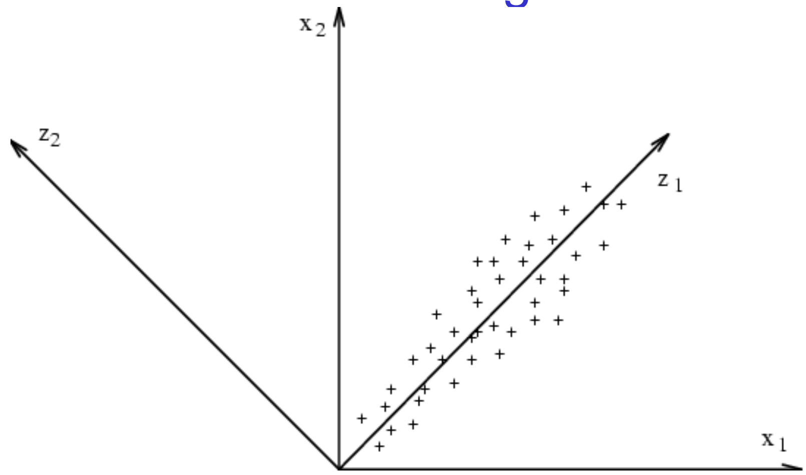


- Observations with two highly-correlated variables:
  e.g. grey-value at neighbouring pixels OR
  length&weight of growing children
- Highly correlated values: x_1 has info on x_2
- Instead of storing 2 variables, we can store only 1
  (needs also the relation of this to original variables, i.e. PCA)

Correlation knowledge helps in compression, inspection, and classification

Computer Vision

Decorrelation through rotation

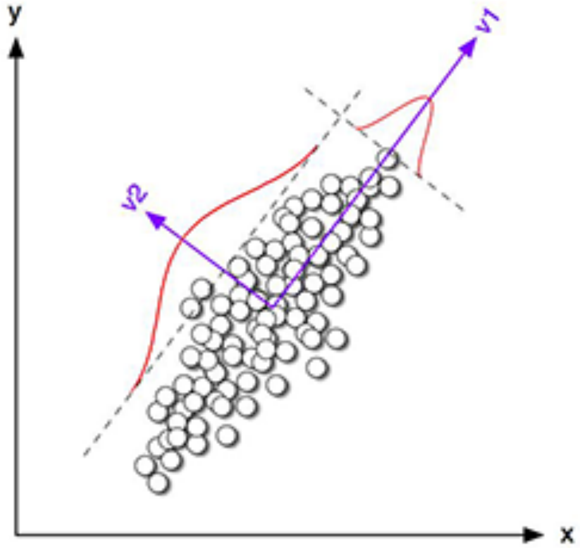


- Using correlation, rotate frame axes, s.t. maximize variation in 1st component, minimize in 2nd
- We can then potentially drop z_2 now

Recall the principle behind *unitary transforms* : rotation in high dimensional spaces

Computer Vision

Decorrelation through rotation



We shall work around the mean:

- Thus, we are applying a rotation about the mean of the distribution (analogous to ellipse fitting)
- Extends to hyperellipsoids in higher dimensions, where visual inspection is not possible

Decorrelation through rotation

Sum of variances do not change with rotations:

$$\sum_{i=1}^{p} \sigma_i^2 = \sum_{j=1}^{p} \tilde{\sigma}_j^2$$

With σ_i^2 variance in x_i and $\tilde{\sigma}_j^2$ variance in z_j

result of invariance of center of gravity and distance under rotation

Parseval equation

redistribution of energy / variance

We want as much variance in as few coordinates

PCA: introduction

In high dimensional spaces, an optimal rotation no longer clear upon visual inspection

Some statistics needed: covariance matrix (note: underlying assumption of Gaussian distr.!)

Intuitive: fit hyperellipsoid to cluster subsequent PCs correspond to axes from the longest to the shortest

Computer Vision

PCA: method

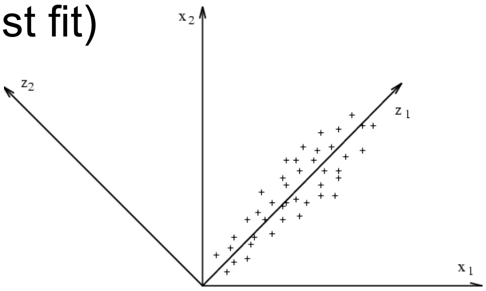
Suppose x is a vector of p random variables

(can extend to points in space & images with pixels)

first step: look for a linear combination $c_1^T x$ which has maximum variance (fitting a line in R^N)

second step: look for a linear combination, $c_2^T x$ uncorrelated (orthogonal) with $c_1^T x$ and with maximum variance (best fit)

third step: repeat...



Computer Vision

Algorithm : Find PCA basis formally

1. Consider $c_1^T x$ with c_1 and maximize its variance

var [$c_1^T x$] =

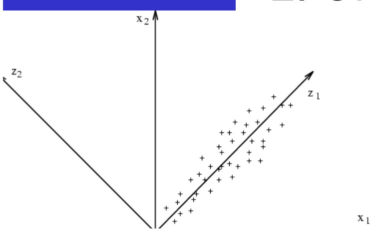
$$\sum c_1^T x \; (c_1^T x)^T = \sum c_1^T x \, x^T c_1 \; = c_1^T \sum (x \, x^T) c_1$$

$$= \; c_1^T C \, c_1 \quad \text{is maximized,}$$

where C is the covariance matrix

(assuming data is centered around its mean)

2. Orthonormality: for a unit norm vector: $c_1^T c_1 = 1$



Computer Vision

PCA algorithm: c_1

Using Lagrange multipliers we maximize

$$c_1^T C c_1 - \lambda\left(c_1^T c_1 - 1\right)$$

Differentiation w.r.t. c_1 gives

$$C c_1 - \lambda c_1 = 0$$

$$\left(C - \lambda\, I_p\right) c_1 = 0$$

where I_p is the (p x p) identity matrix

Thus, λ must be an eigenvalue of C,
where c_1 is the corresponding eigenvector

Computer Vision

PCA algorithm: c_1

Which of the p eigenvectors?

$$c_1^T C c_1 = c_1^T \lambda c_1 = \lambda c_1^T c_1 = \lambda$$

So λ must be as large as possible

Thus, c_1 is the eigenvector with the largest eigenvalue

The k^{th} PC is the eigenvector
with the k^{th} largest eigenvalue

Computer Vision

PCA algorithm: c_2

Proof for k = 2

Maximize $c_2^T C c_2$ while uncorrelated with $c_1^T x$

$$\operatorname{cov}\left[c_1^T x, c_2^T x\right] =$$

$$c_1^T C c_2 = c_2^T C c_1 = c_2^T \lambda_1 c_1 = \lambda_1 c_2^T c_1 = \lambda_1 c_1^T c_2$$

Thus uncorrelatedness becomes

$$c_1^T C\, c_2 = 0,\; c_2^T C\, c_1 = 0,\; c_1^T c_2 = 0,\; c_2^T c_1 = 0$$

Computer Vision

PCA algorithm: c_2

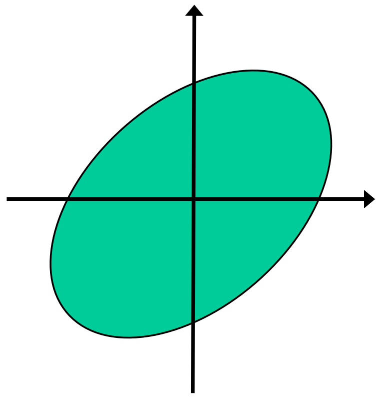
decorrelation vs. orthogonality

$$c_1^T C\, c_2 = 0,\; c_2^T C\, c_1 = 0,\; c_1^T c_2 = 0,\; c_2^T c_1 = 0$$

go hand in hand only for main axes of the ellipsoid defined by the covariance matrix !

Computer Vision

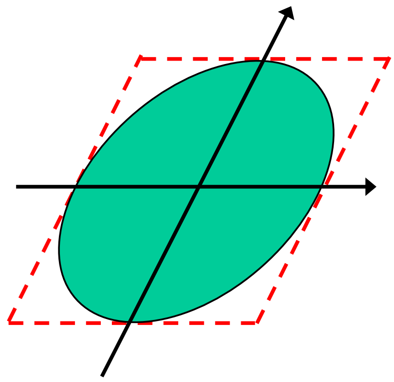
PCA: Decorralation vs. orthogonality example



The given axes are orthogonal,

But max decorrelation is not achieved

The conjugate axis would yield max decorrelation, but is not orthogonal now



To satisfy orthogonality,
the most decorrelated axis should be picked
among orthogonal ones to the first one

Computer Vision

PCA algorithm: c_2

Then, using λ, ϕ as Lagrange multipliers

$$c_2^T C c_2 - \lambda \left(c_2^T c_2 - 1 \right) - \phi \, c_2^T c_1$$

Differentiation w.r.t. c_2 gives

$$C c_2 - \lambda c_2 - \phi c_1 = 0$$

Multiplication on the left by c_1^T gives

$$c_1^T C c_2 - \lambda c_1^T c_2 - \phi \, c_1^T c_1 = 0$$

Thus ϕ = 0

Therefore, $C c_2 - \lambda \, c_2 = 0$, i.e. $\left(C - \lambda I_p \right) c_2 = 0$

Again, maximize $c_2^T C c_2 = \lambda$, so select 2nd largest λ_2

Computer Vision

PCA: interpretation

Similarly, the other PCs can be shown to be eigenvectors of C corresponding to the subsequently next largest eigenvalues

Because C is a real, symmetric matrix, we know all its eigenvectors will be orthogonal

We therefore can interpret PCA as a coordinate rotation/reflection in a higher dimensional space (orthogonal transformation)

Computer Vision

Decorrelation through rotation

Principal Component Analysis (PCA): collects maximum variance in subsequent uncorrelated components.

In that sense, it is the optimal rotation.

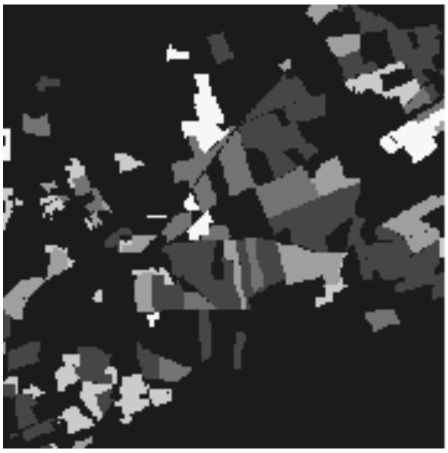
PCs can be interpreted as linear combinations of original variables.

Strongly correlated data $\Rightarrow$ first PCs contain most of the variance
information loss is minimal if only retaining these

Classification example with PCA: satellite images

Example: classification of 5 crop types

Input: 3 spectral bands from SPOT satellite
Near-infrared (N), Red (R), and Green (G)
each pixel = 20 m x 20 m

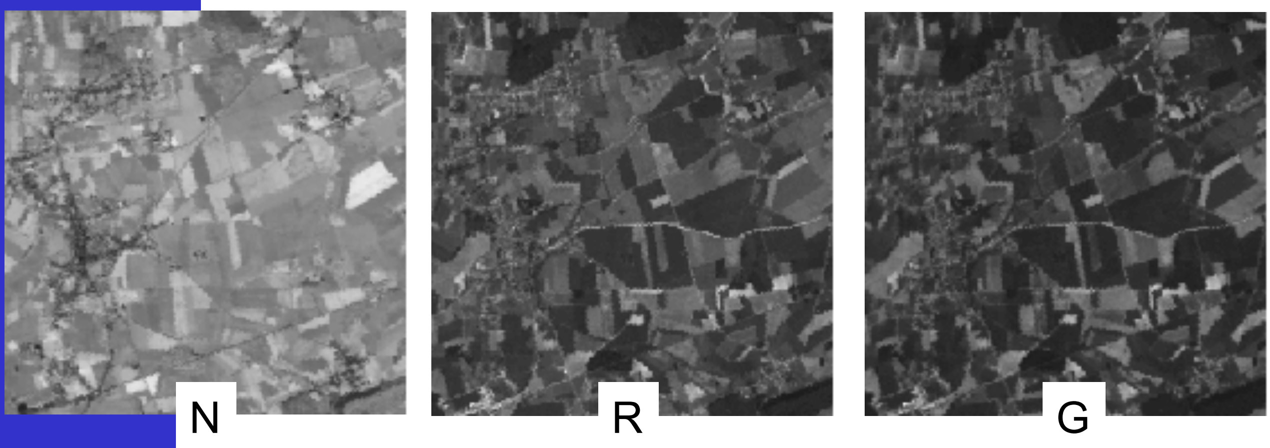


Comparison of 2 PCs vs. 3 original bands

Classification example : satellite images

Until now, each image was a sample,
with a dimension of #pixels

In this example, each pixel is a sample,
with a dimension of #colors (i.e. 3).



Computer Vision

Classification example : satellite images

Observation: correlation between R and G
N seems uncorrelated

Given all pixels of this sample image,
a 3x3 covariance matrix of “colors” can be found:

$$C = \begin{pmatrix} 127.2447 & 13.3062 & -5.9095 \\ 13.3062 & 34.2264 & 39.2092 \\ -5.9095 & 39.2092 & 54.8805 \end{pmatrix}$$

Which corroborates the observation,
as for correlation coefficient computed by $\sigma_{ij} = \frac{c_{ij}}{\sigma_i \sigma_j}$

$\sigma_{NR} = 0.2016$ and $\sigma_{RG} = 0.9047$

Computer Vision

Classification example : satellite images

Principal Components:

$$\begin{pmatrix} 0.9907 \\ 0.1360 \\ -0.0070 \end{pmatrix} \begin{pmatrix} -0.0765 \\ 0.5980 \\ 0.7978 \end{pmatrix} \begin{pmatrix} -0.1127 \\ 0.7899 \\ -0.6028 \end{pmatrix}$$

with eigenvalues: 129.1135, 84.8359, and 2.4022

1^{st} PC ≈ near-infrared input N
2^{nd} PC ≈ a combination of R&G

notice the low eigenvalue of 3rd PC,
which we can thus ignore

Classification results would then compare:
3 original bands: 76.3 % accuracy
2 first PCs: 73.5 % accuracy
Note only a very minor accuracy loss

Computer Vision

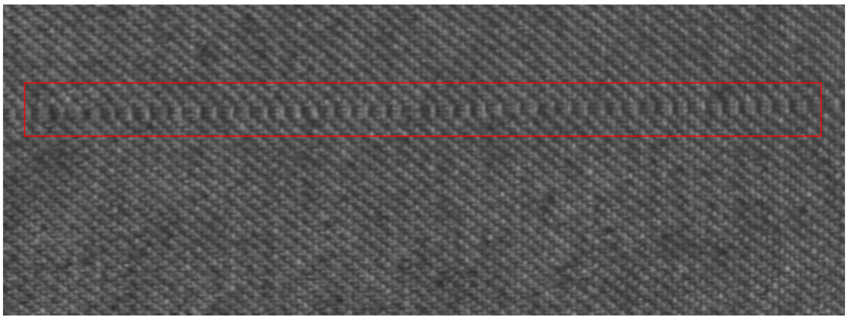
Inspection ex.: eigenfilters for textile

Motivation:

> High eigenvalues indicate eigenvectors (filters) representing ordinary, repeating, common patterns.

> Conversely, lower eigenvalues (or differences between responses to different filters) may help detect out-of-ordinary, rare occurrences.

Example application: textile inspection



Filters are applied with size of one period [8,6]

(period found as the peak in autocorrelation)

Computer Vision

Inspection ex.: eigenfilters for textile

(Complete example in Texture lecture)

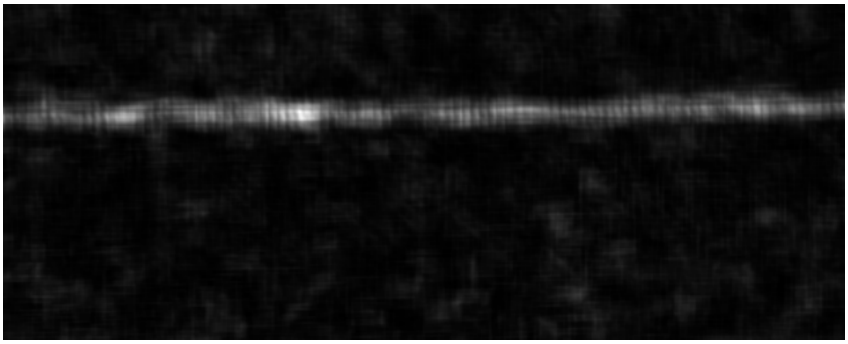
As we will see, PCA allows for the design of dedicated convolution filters, ordered by the variance in their output when applied across the image.

Flaws which won't follow the typical pattern may then express itself in low-variance components or variation across filter responses (as outlier values).

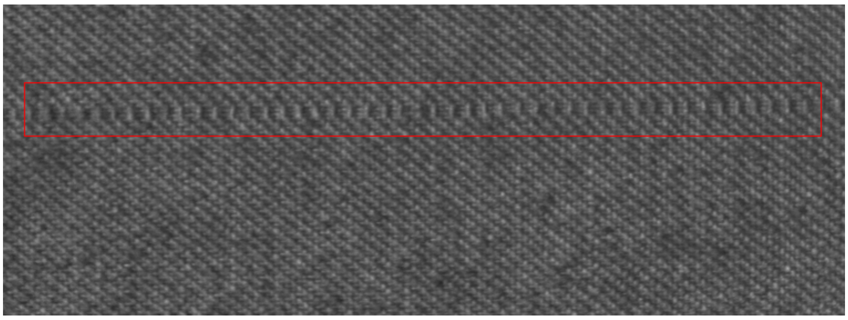
Computer Vision

Inspection ex.: eigenfilters for textile

Mahalanobis distance of filter energies:



Flaw region found by thresholding:



Computer Vision

Image compression ex.: eigenfaces

Averaging of input faces

"Mean" face

Image compression ex.: eigenfaces

Neighbouring pixel intensities are highly correlated

Consider image as large intensity vector

Eigenvectors: *"eigenimages"*

Computational problems :
N^2 x N^2 covariance matrices!

Specifying image statistics: which exemplary set?

Image dependence: eigenimages needed!

Image compression ex.: eigenfaces

Karhunen-Loève transform = PCA on images

Redistributes variance over a few components most efficiently

Best approximation: Minimal least-square error for truncated approximations

Dimensionality problem can be remedied: formulation as eigenvalue problem in space of dimension equal to number of sample images

Computer Vision

Dimension in number of samples/images

n samples, p-dimensional space, $n < p$

Consider the $(p \times n)$-matrix X with samples as columns

(p x p) covariance matrix

$$C = \frac{1}{n} X \; X^T$$

Much smaller (n x n) matrix

$$S = X^T X$$

$$X^T X \; c_i = \lambda_i c_i$$

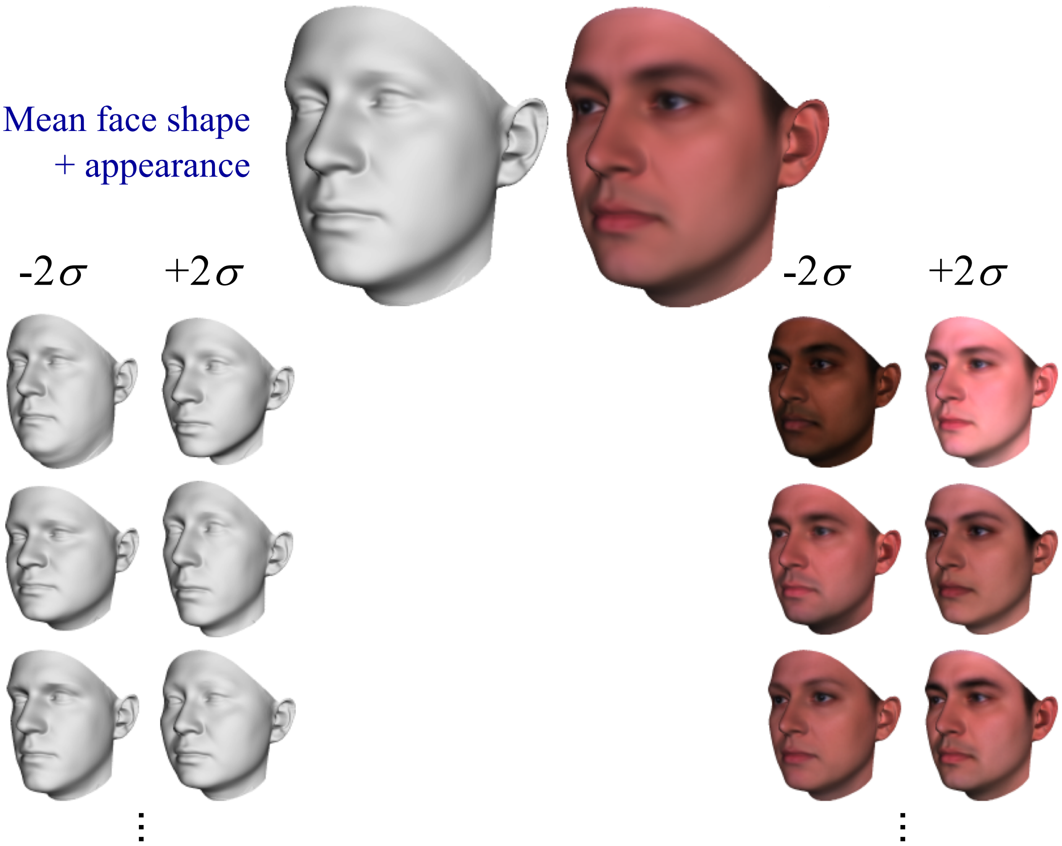
$$X \; X^T X \; c_i = X \; \lambda_i c_i$$

$$\left(\frac{1}{n} X \; X^T\right)\left(X \; c_i\right) = \left(\frac{\lambda_i}{n}\right)\left(X \; c_i\right)$$

Eigenvectors of a (n x n)-matrix need to be found

Computer Vision

Shape can also be PCA'ed, e.g.



Variation in shape

Variation in appearance

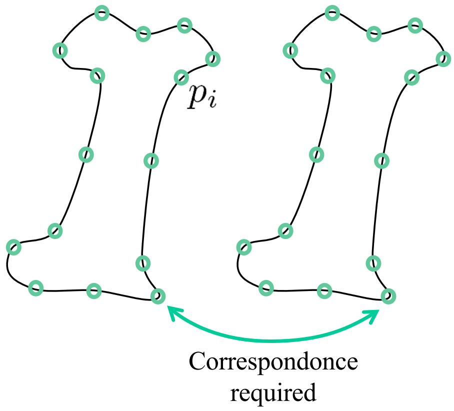
Computer Vision

Statistical Shape Modeling

IDEA: If the sought shape is known,
use that to analyze shapes or regularize a surface fitting

<u>Point Distribution Model</u>
Shapes as a set of points:

$$v_i = (\varphi_i(p_1), \ldots, \varphi_i(p_N)) \in \mathbb{R}^{N \cdot d}$$



Shape Modeling
with Principal Component Analysis (PCA):

$$\overline{m} = \frac{1}{n} \sum_{i=1}^{n} v_i$$

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (v_i - \overline{m})(v_i - \overline{m})^T$$ Covariance matrix

Generative Model for (similar) shapes:

$$v = v(\alpha_1, \ldots, \alpha_m) = \overline{m} + \sum_{i=1}^{m} \alpha_i \lambda_i u_i$$

$$\alpha \sim \mathcal{N}(0, I_m) \qquad v \sim \mathcal{N}(\overline{m}, UD^2U^T) \approx \mathcal{N}(\overline{m}, S)$$

Variation around mean shape

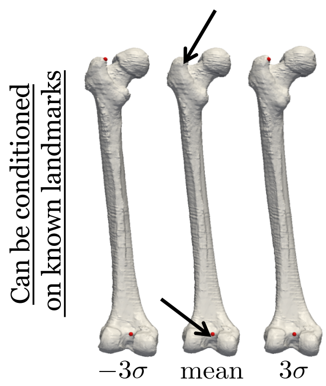
Computer Vision

SSMs for segmentation (also comes later)

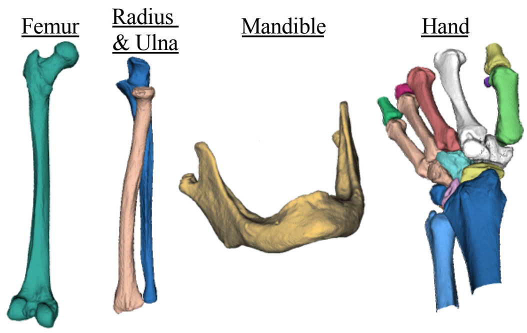
Can be “trained” from example shapes:
i.e. find the covariance matrix after aligning shapes

Fitted iteratively to shape edges, as in deformable contours
(in contrast, fitting move is projected onto shape [PCA] space)

Image (edge) appearance at shape nodes can also be modeled
in order to use in the iterative fitting process
→ “Active Shape and Appearance Models” ASM / AAM



Segmentation examples



Computer Vision

Independent Component Analysis

Computer Vision

Goal of ICA

Suppose we have *n* signals/images *i* , which are linear combinations of *n* underlying signals/images *u*

$$\mathrm{i} = A\mathrm{u}$$

ICA aims to extract the u_i

Examples for ICA

3 sound sources at different positions in a room, captured by 3 microphones, also at different positions.

The 3 microphones would capture 3 different linear mixes.
ICA can – from the 3 microphone signals – deduce the 3 original sounds.

A vision example is to extract a pattern behind a window and a pattern reflected in it, if 2 images were taken under different illuminations, such that the relative amounts of both are different.

Unmixing window reflection & background

2 images

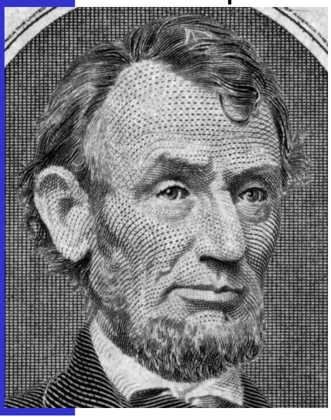
ICA components

Fingerprint on a banknote (a 2nd example)

Forensic Sample

Clean Banknote (registered)

Independent components



Contrast enhanced

Remarks on ICA

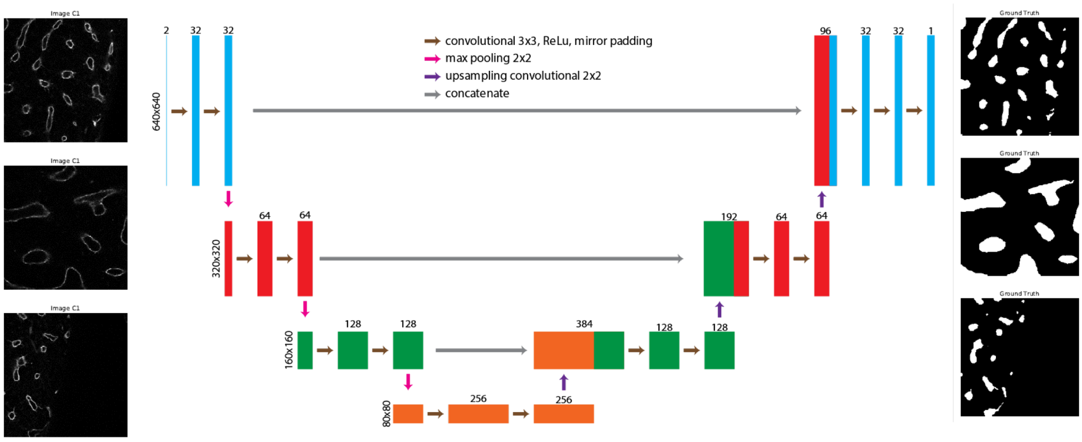
ICA is not a unitary transformation, i.e. not a rotation!

Instead, it is a general linear transformation.

This is also logical, as it has to apply (the inverse of) an arbitrary linear transformation.

The algorithm is based on the assumption that the underlying signals *u* are statistically independent (and not just decorrelated as with PCA).

Convolutions in Neural Networks:



Coarsening structure (U-net) : **scale-space**
Convolutions: image transformations
(not necessarily unitary)

Needs “training” examples (to tune transformations)
Still aims to project data in a (feature) space,
where it is most discriminative / best separable.
sounds familiar to PCA?

Image compression ex.: eigenfaces

Modes **Morphs**