Lecture 8

Images as n-Dimensional Vectors
Reminders

Midterm is next week, here at 6:00.
   Please be here on time.

A2 is due this Sunday (March 2) at 11:59:59.999.

Alternative office hour on Mondays at noon.
Topic 05:

Representing Images as n-Dimensional Vectors

- Template matching:
  - cross-correlation & normalized cross-correlation
- Principal component analysis
  - geometrical intuition: changing basis
  - the eigenfaces recognition algorithm
- algorithm derivation: minimizing sample covariance
Template Matching Applications

Face detection & recognition

Fujifilm Debuts FinePix Digital Camera With Face Detection Technology

New FinePix S6000fd offers breakthrough focusing technology, company also rolls out compact FinePix F20.

BY: POPS PHOTO.COM STAFF
July 12, 2006

Fujifilm introduces the SLR-styled FinePix S6000fd, the first digital camera in Fujifilm’s line-up with the company’s revolutionary new Face Detection Technology.

Face Detection Technology operates exactly as its name implies, identifying up to 10 faces in a framed scene. Once faces are identified and prioritized, the 6.3-megapixel FinePix S6000fd adjusts its focus and exposure accordingly to ensure the sharpness and clarity of human subjects in the picture, regardless of background. And since it is hardware rather than software based, Fujifilm’s Face Detection Technology works in as little as 0.35 seconds, faster than similar in-camera detection systems currently on the market or soon to be available.

Quicker operation is said to reduce the likelihood of missed or blurry photos, frustrations often associated with digital photography. The advanced Face Detection Technology system built into Fujifilm’s new FinePix S6000fd digital camera is based on the image intelligence technology found in Fujifilm’s Frontier Digital Lab Systems, used by photo finishers to produce...
Representing Images & Patches as Vectors

As graph in 2D

We covered some tools that can be applied to 1-D image patches represented as vectors.
Representing Images & Patches as Vectors

As graph in 2D

For example, a patch of radius \( w \):

\[
\text{Patch} \begin{bmatrix} x_1 & x_2 & \cdots & x_{2w+1} \end{bmatrix} \text{ (2w+1 pixels)}
\]

is simply a \((2w+1)\)-dimensional (column) vector.
Representing Images & Patches as Vectors

As graph in 2D

And you can think of it as a point in the \((2w+1)\)-dimensional space.
Representing Images & Patches as Vectors

A simple example is a patch $x$ that is just a 2-dimensional vector, such as a 2 image pixel patch:

\[
\begin{bmatrix}
50 \\
255
\end{bmatrix}
\]

which can be represented as a point in 2D space.
Representing Images & Patches as Vectors

And if we think in terms of 2-pixel patches, a tiny (1x10 pixels) image like:

| 50 | 255 | 30 | 80 | 200 | 110 | 50 | 200 | 280 | 100 |

Can be thought of as containing nine 2D patches (vectors)

\[ x_1, x_2, x_5, x_9 \]

With their corresponding 2D point representation:

Intensity of pixel 1

Intensity of pixel 2
Representing Images & Patches as Vectors

Now, if patches are size 3, then a patch like

\[
\begin{bmatrix}
50 & 255 & 30
\end{bmatrix}
\]

can be represented as a point in 3D space.
And the same tiny (1x10 pixels) image can be thought of as containing eight (not nine) 3D vectors with their corresponding 3D point representation:
But the choice of patch size looks pretty arbitrary...
But the choice of patch size looks pretty arbitrary...

And it is!

We determined which pixels go in each patch.
Representing Images & Patches as Vectors

Is there anything preventing us from creating 1D vectors from 2D patches then?

Think of the following 3x10 image:

Can we think of this patch as a vector of size 9?
Is there anything preventing us from creating 1D vectors from 2D patches then?

Think of the following 3x10 image:

\[ X_i = \begin{bmatrix} 80 \\ 30 \\ 100 \\ 50 \\ 60 \\ 30 \\ 30 \\ 60 \\ 30 \\ 90 \end{bmatrix} \]
Is there anything preventing us from creating 1D vectors from 2D patches then?

Think of the following 3x10 image:

Absolutely!

Equally arbitrary, but similar properties and interpretation.
Now, why would we want to lose the absolute spatial information?
Because a vector representation allows us to compute similarity between (1D or 2D) patches using simple vector operations.

Dissimilar

Similar
Representing Images & Patches as Vectors

Patch similarity is the foundation of an important detection procedure called template matching. The intuition is, can we find the location of this template in this image?
Representing Images & Patches as Vectors

And get this location as the one most similar?
Representing Images & Patches as Vectors

Template matching sounds useful but...

How?

At what computational cost (in terms of memory and number of operations)?
Representing Images & Patches as Vectors

Estimating similarity between image patches. Here are the patches from 3 slides ago, in 2D space:

\[ x_1 = [100, 200] \]
\[ x_2 = [200, 50] \]

\[ x_1 = [100, 200] \]
\[ x_2 = [180, 190] \]

Dissimilar

Similar
The Template Matching Problem

The goal is to find the image patch $x_i$ that is most similar to a template $T$.
How about the distance?
The Template Matching Problem

Measuring distances can be done in many different ways. Here is our Similarity function #1.

Root Mean Squared Distance

\[ \text{RMS}(x_i, T) = \| x_i - T \| \]

\[ = \left[ (x_i - T)^T(x_i - T) \right]^{1/2} \]
The Template Matching Problem

The goal is to find the image patch $x_i$ that is most similar to a template $T$. The problem can be formally written as:

$$\text{Find } \arg\min_{x_i} \|x_i - T\|$$

argmin is a shorthand for “the $x_i$ that minimizes the expression to the right”.
The Template Matching Problem

The goal is to find the image patch $x_i$ that is most similar to a template $T$. The problem can be formally written as:

$$\text{Find} \quad \arg\min_{x_i} \| x_i - T \|$$

Note that efficiency can be improved by minimizing $\| x_i - T \|^2$ which minimizes in the same $x_i$ and saves the square root computations. This new formulation can then be written as:

$$\arg\min_{x_i} (x_i - T)^T(x_i - T)$$
The Template Matching Problem

Again, note that this 1D metric is equivalent to the 2D operation that keeps the spatial relation of the template and the image.

For instance, if a patch is centered at pixel \((r,c)\) and the template is of radius \(N\), then the following equation computes the RMS distance between the image patch and the template.

\[
\text{rms\_dist}(r,c) = \sqrt{\sum_{a=-N}^{N} \sum_{b=-N}^{N} (I(r+a, c+b) - T(a, b))^2}
\]
Template Matching Algorithm

Basic Template Matching Algorithm:

1. Define a matrix $\text{RMS}_\text{Dist}$ of size equal to the image. This will hold the RMS distance at each pixel.
2. Compute $(\chi_i - \omega)^T(\chi_i - \omega)$ for each patch $\chi_i$ (centered at coordinates $(c, r)$ in the image), where it is possible to compute.
3. When RMS distances between the template and all patches have been computed, search over $\text{RMS}_\text{Dist}$ to find the pixel with lowest intensity.
Template Matching Algorithm

Basic Template Matching Algorithm:

1. Define a matrix $\text{RMS\_Dist}$ of size equal to the image. This will hold the RMS distance at each pixel.
2. Compute $(\chi_i - \tau)^T(\chi_i - \tau)$ for each patch $\chi_i$ (centered at coordinates $(c, r)$ in the image), where it is possible to compute.
3. When RMS distances between the template and all patches have been computed, search over $\text{RMS\_Dist}$ to find the pixel with lowest intensity.

What is the problem with this distance metric?
The problem with this distance metric.

\[ \left( (x_i - \tau)^T (x_i - \tau) \right)^{1/2} \]

Thoughts?
Template Matching Algorithm

The problem with this distance metric.

What is the distance from T to this new patch x?
Template Matching Algorithm

The problem with this distance metric.

The same!
Template Matching Algorithm

The problem with this distance metric.

Circle of vectors that are equidistant from $T$

In fact, there is an infinite number of image patches that render the same distance.
Template Matching Algorithm

But there is a problem with this distance metric.

In fact, there is an infinite number of image patches that render the same distance.
But there is a problem with this distance metric.
Representing Images & Patches as Vectors

\[
\text{RMS of } \begin{array}{c} \text{in} \\ \end{array} \begin{array}{c} \text{would be small} \\ \end{array} \\
\begin{array}{c} \text{I}_1 \\ \end{array}
\]

\[
\text{RMS of } \begin{array}{c} \text{in} \\ \end{array} \begin{array}{c} \text{would be large} \\ \end{array} \\
\begin{array}{c} \text{I}_2 \\ \end{array}
\]
Template Matching Algorithm

$I_2$ is just a scaled version of $x_1$, but the $\text{RMS}(x_2,T)$ is much bigger than $\text{RMS}(x_1,T)$ (which is almost zero)
Representing Images & Patches as Vectors

RMS of \( T \) in \( \text{in} \) would be small

RMS of \( T \) in \( \text{in} \) would be large

RMS cannot distinguish between patches that are just scaled versions of \( T \), from other patches that differ in other ways.
Representing Images & Patches as Vectors

Is there anything we can do?
Topic 05:
Representing Images as n-Dimensional Vectors

- Template matching:
  - cross-correlation & normalized cross-correlation
- Principal component analysis
  - geometrical intuition: changing basis
  - the eigenfaces recognition algorithm
  - algorithm derivation: minimizing sample covariance
The Template Matching Problem

Measuring distances can be done in many different ways. Similarity function #2.

Cross-correlation

$$\text{CC} \left( x_i, T \right) = x_i^T \cdot T$$

i.e. the dot product between the two vectors

The projection from one onto the other.
The Template Matching Problem

Properties of cross-correlation $CC(x_i, T)$

- Depends on the lengths of $x_i$ and $T$ (still 😞)

- From all $x_i$ of the same length, the CC is maximum when $x_i$ and $T$ have the same direction.

- CC is zero when $x_i$ and $T$ are orthogonal (most dissimilar!)

$$CC \left( x_i, T \right) = X_i^T \cdot T$$

$$= \|X_i\| \cdot \|T\| \cdot \cos \Theta$$
The Template Matching Problem

Measuring distances can be done in many different ways. Similarity function #3.

Normalized Cross-Correlation

\[ \text{NCC} \ (X_i, T) = \frac{X_i^T \cdot T}{\|X_i\| \cdot \|T\|} = \cos \theta \]

The cosine of the angle between the two vectors.
The Template Matching Problem

Properties of Normalized Cross-Correlation:

Independent of lengths of \( x_i \) and \( T \) (finally 😊)

Maximum (\( \text{NCC}(x_i, T) = 1 \)) when the intensities of \( x_i \) and \( T \) are the same, up to a scale factor

Minimum (\( \text{NCC}(x_i, T) = 0 \)) when \( x_i \) and \( T \) are orthogonal (most dissimilar).

\( \text{NCC}(x_i, T) = \text{CC}(x_i, T) \) when \( x_i \) and \( T \) are unit vectors.
2D Template Matching Using CC & NCC

Note that Cross-Correlation and Normalized Cross-Correlation can be computed as a 2D sum

$$\text{cc-dist} (r,c) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} I(r+a, c+b) \cdot T(a, b)$$

<table>
<thead>
<tr>
<th>Row 0</th>
<th>Row r</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 255 90</td>
<td>80 200 100</td>
</tr>
<tr>
<td>80 200 100</td>
<td>80 90 100</td>
</tr>
<tr>
<td>150 90 30</td>
<td>80 90 100</td>
</tr>
</tbody>
</table>

Q: What is the 2D sum expression for NCC?
Applying this procedure to the entire image

Template $T$ (M pixels)

$T \rightarrow [\quad ]$

$T = M$ -dimensional column vector

Image $I$ (N patches)

$I \rightarrow [\quad ]$

Each image patch = $M$ -dimensional column vector

What is the computational complexity?
2D Template Matching Using CC & NCC

For instance match a template of $21 \times 21$ to an image of $1000 \times 1000$

Template $T$ ($M$ pixels)

\[
\frac{1}{4} \begin{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \end{bmatrix}
\]

$T = M$ -dimensional column vector

Image $I$ ($N$ patches)

\[
\frac{1}{1000} \begin{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \end{bmatrix}
\]

$N \approx 10^6$

Each image patch = $M$ -dimensional column vector
2D Template Matching Using CC & NCC

For instance match a template of $21 \times 21$ to an image of $1000 \times 1000$.

If we use CC as the distance metric, we do $M$ multiplications and $M-1$ additions per pixel in the image $I$. 
2D Template Matching Using CC & NCC

For instance match a template of $21 \times 21$ to an image of $1000 \times 1000$

The complexity when using CC is in the order of $O(MN)$ operations for the entire image.
2D Template Matching Using CC & NCC

For instance match a template of 21x21 to an image of 1000x1000

The complexity of NCC is also O(MN), with only some more products and additions to normalize the patch \(x_i\) vectors.
2D Template Matching Using CC & NCC

For instance match a template of $21 \times 21$ to an image of $1000 \times 1000$

These are over 1 billion operations!
Matching a template of \(21x21\) to an image of \(1000x1000\) requires around \textbf{1 billion} operations.

Is there a way to represent \(x_i\) and \(T\) with \(d<<M\) to improve efficiency?

Taking \(O(MN)\) down to \(O(dN)\)

(with \(d = 5\), as opposed to \(d = 441\), for instance)
This problem is called Dimensionality Reduction

and using it can lead to speed-ups of orders of magnitude!
Template Matching: Computational Issues

Demo!
Topic 05:

Representing Images as n-Dimensional Vectors

- Template matching:
  - cross-correlation & normalized cross-correlation
- Principal component analysis
  - geometrical intuition: changing basis
  - the eigenfaces recognition algorithm
  - algorithm derivation: minimizing sample covariance
Let's look at an example to develop some intuition about dimensionality reduction.

Imagine a set of 2-pixel patches whose $x_1$ and $x_2$ values are uncorrelated.
Linear Dimensionality Reduction: Basic Intuition

Let's look at an example to develop some intuition about dimensionality reduction.

The coordinate of each of these patches can be determined given two basis vectors.
Linear Dimensionality Reduction: Basic Intuition

Let's look at an example to develop some intuition about dimensionality reduction.

I chose unit vectors that aligned with the x and y axis, but any two (non-parallel) vectors could have been used.
In this setting, a point $x_i$ can be written as:

$$x_i = \begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_i^1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_i^2$$
And the CC \((T, x_i)\) as

\[
\text{CC}(T, x_i) = T^T \cdot x_i \quad \text{unconstrained}
\]

\[
= T^T \left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_i^1 + \begin{bmatrix} 0 & 1 \end{bmatrix} x_i^2 \right)
\]

\[
= \left( T^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) x_i^1 + \left( T^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) x_i^2
\]
And the CC $\langle T, x_i \rangle$ as

$$CC(T, x_i) = T^T \cdot x_i \quad \text{unconstrained}$$

$$= T^T \left( \begin{bmatrix} 1 & 0 \end{bmatrix} x_i^1 + \begin{bmatrix} 0 & 1 \end{bmatrix} x_i^2 \right)$$

$$= \left( T^T \begin{bmatrix} 1 & 0 \end{bmatrix} \right) x_i^1 + \left( T^T \begin{bmatrix} 0 & 1 \end{bmatrix} \right) x_i^2$$

Notice both $x_i^1$ and $x_i^2$ are relevant.
Linear Dimensionality Reduction: Basic Intuition

Now imagine that pixels in a patch are correlated.

Let’s not imagine, but look at some actual data!
Linear Dimensionality Reduction: Basic Intuition

If pixel intensities are correlated, as in:

![Scatter plot showing correlated pixel intensities](image-url)
Then we can use different basis vectors with interesting properties, for instance assume the basis vectors in black.
Now, note that when pixel intensities are correlated, it is possible to express a patch in terms of basis vectors where only a few of the coordinates are significant (not close to zero):

\[ x_i = B_1 y_i^1 + B_2 y_i^2 \]

Close to zero
Linear Dimensionality Reduction: Basic Intuition

\[ x_i = B_1 y_i^1 + B_2 y_i^2 \]

Close to zero

And when this is true, then:

\[ x_i = y_i^1 B_1 + y_i^2 B_2 \approx y_i^1 B_1 \]

\[ CC(T, x_i) = T^T X_i \]

\[ = y_i^1 (T^T B_1) + y_i^2 (T^T B_2) \approx y_i^1 (T^T B_1) \]

unconstrained near 0
Linear Dimensionality Reduction: Basic Intuition

\[ x_i = B_1 y_i^1 + B_2 y_i^2 \]

Close to zero

And when this is true, then:

\[
\text{CC}(T, x_i) \approx y_i^1 (T^T B_1)
\]

Compared to:

\[
\text{CC}(T, x_i) = (T^T \begin{bmatrix} 1 & 0 \end{bmatrix}) x_i^1 + (T^T \begin{bmatrix} 0 & 1 \end{bmatrix}) x_i^2
\]

Note that when pixels intensities are related, the choice of basis vectors can make a big difference in computational complexity.
In summary:

We now know that carefully chosen basis vectors can represent image patches of correlated pixels much more efficiently.

And we also know that in “natural images”, pixel intensities inside each patch are highly correlated.

We can exploit these two pieces of knowledge to do template matching much more efficiently.
Algorithm:

1) Find the optimal set of basis vectors $B_1, B_2, \ldots, B_m$. These basis are often called the Principal Components.

2) Compute patch coordinates in that basis

3) Discard the axes with near zero coordinates for all patches.
Changing the Basis: Matrix Notation

Keep in mind that the goal is to go from this

\[ X_i = [0] x_i^1 + [0] x_i^2 \]

To this

\[ x_i = B_1 \cdot y_i^1 + B_2 \cdot y_i^2 \]

(With a pixel grid and axis annotations)
Changing the Basis: Matrix Notation

In the original case, the basis matrix is the identity matrix.

\[ X_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_i' + \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_i^2 \]

In matrix notation (ith patch):

\[ X_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_i' \\ x_i^2 \end{bmatrix} \]

coordinate vector
In the alternative representation, the basis $B_i$ are the transformations that take new coordinates $y_i$, to reconstruct the original data $x_i$.

\[ x_i = B_1 \cdot y_i^1 + B_2 \cdot y_i^2 \]

\[ X_i = \begin{bmatrix} B_1 & B_2 \end{bmatrix} \begin{bmatrix} y_i^1 \\ y_i^2 \end{bmatrix} \]

All $N$ patches

\[ \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix} = \begin{bmatrix} B_1 & B_2 \end{bmatrix} \begin{bmatrix} y_1^1 & y_2^1 & \cdots & y_N^1 \\ y_1^2 & y_2^2 & \cdots & y_N^2 \end{bmatrix} \]
Changing the Basis: Matrix Notation

The same is true for M-Dimensional patches: The reconstructed data $X_i$ is the basis $B$ times the new representations ($Y_i$).

$$
\begin{bmatrix}
X_1 & X_2 & \cdots & X_N
\end{bmatrix}
= 
\begin{bmatrix}
B_1 & B_2 & \cdots & B_M
\end{bmatrix}
\begin{bmatrix}
y_{i1} & y_{i2} & \cdots & y_{iN}
y_{21} & y_{22} & \cdots & y_{2N}
\vdots & \vdots & \ddots & \vdots 
y_{M1} & y_{M2} & \cdots & y_{MN}
\end{bmatrix}
$$
The same is true for M-Dimensional patches: The reconstructed data $X_i$ is the basis $B$ times the new representations ($Y_i$).

\[
\begin{bmatrix}
X_1 & X_2 & \cdots & X_N
\end{bmatrix}
= 
\begin{bmatrix}
B_1 & B_2 & \cdots & B_M
\end{bmatrix}
\cdot
\begin{bmatrix}
y_1^1 & y_2^1 & \cdots & y_N^1 \\
y_1^2 & y_2^2 & \cdots & y_N^2 \\
y_1^d & y_2^d & \cdots & y_N^d \\
y_1^{d+1} & y_2^{d+1} & \cdots & y_N^{d+1} \\
\vdots & \vdots & \ddots & \vdots \\
y_1^M & y_2^M & \cdots & y_N^M
\end{bmatrix}
\]

But crucially, many of these coefficients will be close to zero and can be ignored.
Changing the Basis: Matrix Notation

Eliminating these coefficients leaves us with a d-Dimensional approximation:

\[
\begin{bmatrix}
X_1 & X_2 & \ldots & X_N
\end{bmatrix} = \begin{bmatrix}
B_1 & B_2 & \cdots & B_d
\end{bmatrix} \cdot \begin{bmatrix}
y_1^i & y_2^i & \ldots & y_n^i \\
y_1^d & y_2^d & \ldots & y_n^d
\end{bmatrix}
\]

Note only \(d\) basis are used now, not \(M\)
(and \(d \ll M\))
Finding these (not so) magical Basis in 4 steps:

*Input*: matrix $X$, and desired dimension $d$

*Output*: Basis vectors $B_1, B_2, \ldots, B_d$

1) Compute the average patch

$$\overline{X} = \frac{1}{N} \sum X_i$$

2) Subtract the average patch from each $X_i$

$$Z_i = X_i - \overline{X}$$

3) Define the matrix $Z = [z_1, z_2, \ldots, z_n]$

4) $[B_1, B_2, \ldots, B_d] = $ the eigenvectors of the matrix $ZZ^T$ with the $d$ largest eigenvalues.
Notes on the dimensions of these matrices

The matrix $Z$ is defined as the concatenation of $n$ column-vectors of size $M$, as in:

$$Z = [z_1, z_2, ..., z_n]$$

The size of $Z$ is therefore $[M \times n]$.

The dimension of $ZZ^T$ is $[M \times M]$ (noting that its size is independent of the number of data points ($n$)). So, $ZZ^T$ is square and of size equal to the dimension of one point.

The dimensionality reduction basis $B$ are the first $d$ eigenvectors $B = [b_1, b_2, ... B_d]$ of the matrix $ZZ^T$. The size of $B$ is $[M \times d]$. 