Images as Vectors

Topic 6

Week 5 – Feb. 6^{th}, 2019

Topic 6: 2D Images and Curves

- **Images as Vectors**
- Template Matching
- Cross-Correlation
- Template Matching using Cross-Correlation
- Dimensionality Reduction
- Principle Component Analysis (PCA)
- Understanding PCA
- Case Study: EigenFaces

Images as Arrays

- We are used to seeing images/image patches displayed in 2D
- However, they are stored in memory as 1D arrays:

x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}

Image 2D Array $\mathrm{I}(x, y)$
(5×5)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\dots$	x_{21}	x_{22}	x_{23}	x_{24}

Image 1D Array $X(x)$
(25)

Images as Vectors

- Of course we can also consider the 1D array as a N-dimensional vector, where N is the length of the array, or area of the image:

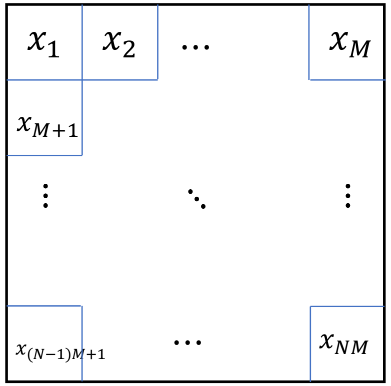


Image Patch $\mathrm{I}(x, y)$
$(N{\times}M)$

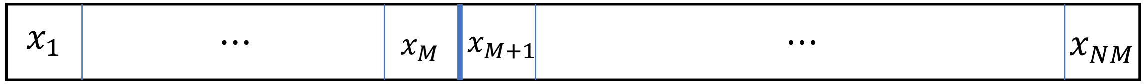
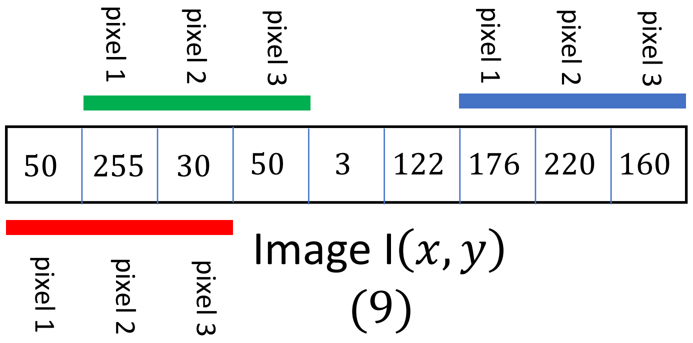


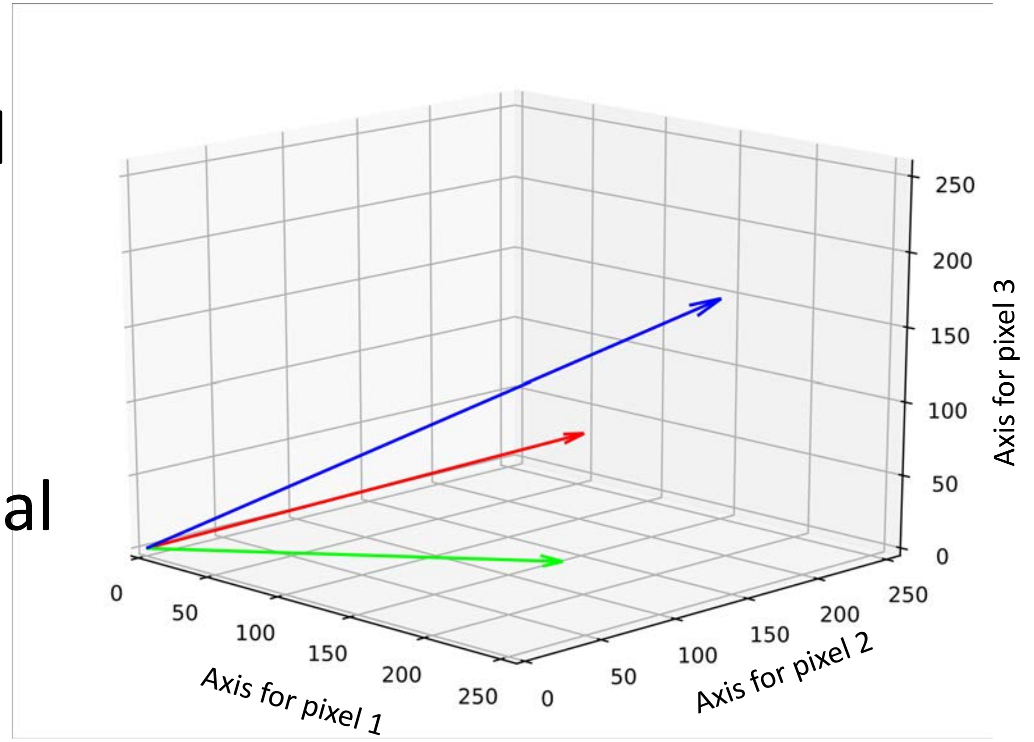
Image Array $X(x)$ (N)

$$\mathrm{I}(x, y) = X(x + M(y - 1))$$

Image Patches as Vectors



- Say we are interested in image patches of dimensions 1×3 from an image of size 1×9
- How many patches can we extract?
  - Imagine sliding 1×3 window
  - [50, 255, 30], [255, 30, 50], …, [176, 220, 160]
  - Get 7 possible patches – due to "boundary"
- Each of these is a vector in a 3D space!
- In general, a patch of size $N \times M$ can be thought of as a point in an NM dimensional space, where **each pixel** is a **different axis**



Takeaway: Images as Vectors

- We can consider any 2D image or image patch as it’s “flattened” array
- For any image/image patch with N rows and M columns, we can also consider this array as a N*M-dimensional vector:

x_1	$\cdots$	x_M	x_{M+1}		$\cdots$				x_{NM}

Image Array $X(x)$
(NM)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \\ \vdots \\ x_{NM} \end{bmatrix}$$

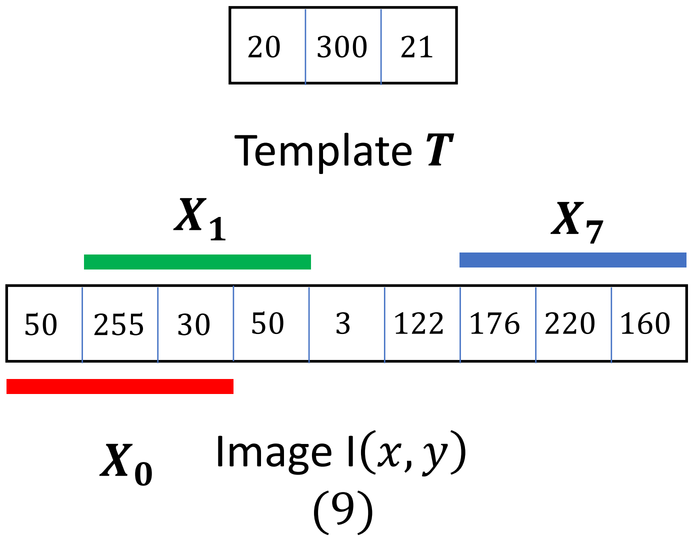
Image Vector $\boldsymbol{X}$

Topic 6: 2D Images and Curves

- Images as Vectors
- **Template Matching**
- Cross-Correlation
- Template Matching using Cross-Correlation
- Dimensionality Reduction
- Principle Component Analysis (PCA)
- Understanding PCA
- Case Study: EigenFaces

Template Matching

- Given a “template” patch $\boldsymbol{T}$ and we want to find an image patch $\boldsymbol{X_i} \in I$ that is **most similar** to our template
- How do we calculate a “similarity” metric?



Similarity Metric #1: Distance

20	300	21

Template $\boldsymbol{T}$

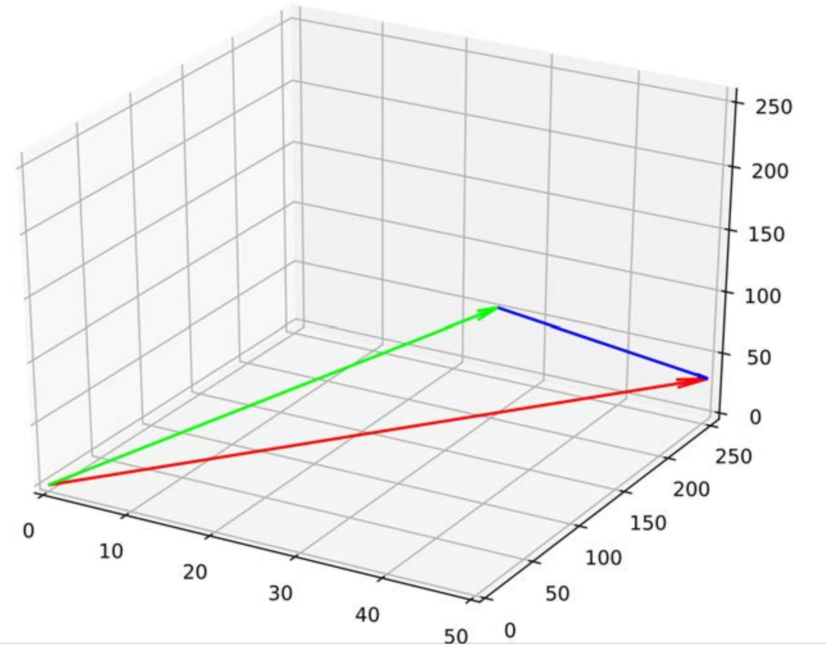
- Given a "template" patch $\boldsymbol{T}$ and we want to find an image patch $x_i \in I$ that is **most similar** to our template
- As we have seen, we can consider our image patches (and template) to be vectors
- Let's try using the Euclidean distance between our two vectors as our similarity metric :

$$\|\boldsymbol{X}_i - \boldsymbol{T}\| = \sqrt{(X_i - T)^T (X_i - T)}$$

  - In some contexts this is known as the root-mean-square (RMS) error

50	255	30	50	3	122	176	220	160

$\boldsymbol{X_0}$ Image I(x, y) (9)



Template Matching using Distance

- Goal: Find image patch $x_i \in I$ that is **most similar** to given template $\boldsymbol{T}$
- Image patch $x_i \in$ is calculated:

$$\text{argmin}_{\boldsymbol{X}_i} ||\boldsymbol{X}_i - \boldsymbol{T}||$$

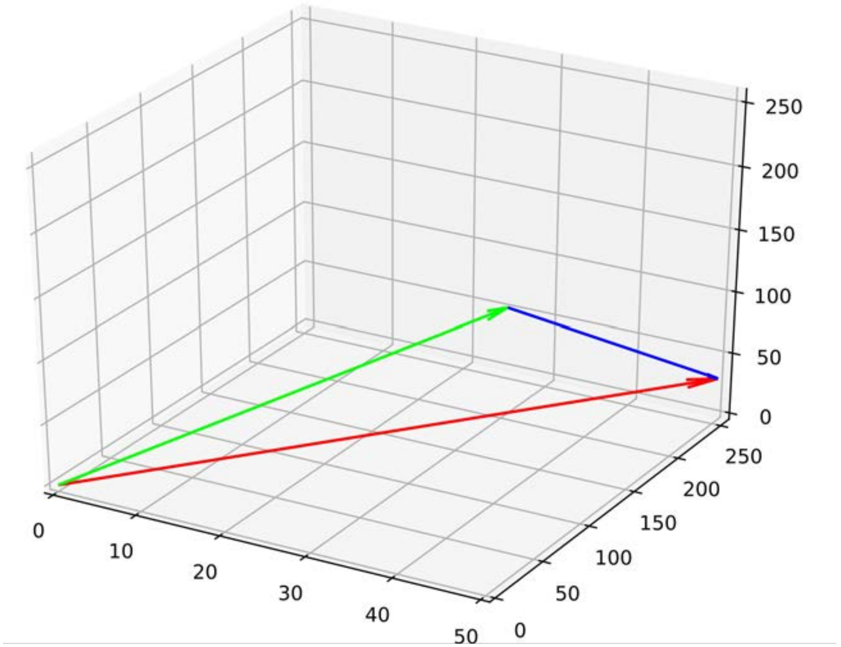
- The notation $\text{argmin}_x\ f(x)$ is shorthand for "give the value of x that minimizes $f(x)$"

20	300	21

Template $\boldsymbol{T}$

50	255	30	50	3	122	176	220	160

$\boldsymbol{X_0}$ Image I(x, y)
(9)



Template Matching using Distance

20	300	21

Template $\boldsymbol{T}$

$$\|\boldsymbol{X}_i - \boldsymbol{T}\| = \sqrt{(\boldsymbol{X}_i - \boldsymbol{T})^T(\boldsymbol{X}_i - \boldsymbol{T})}$$

50	255	30	50	3	122	176	220	160

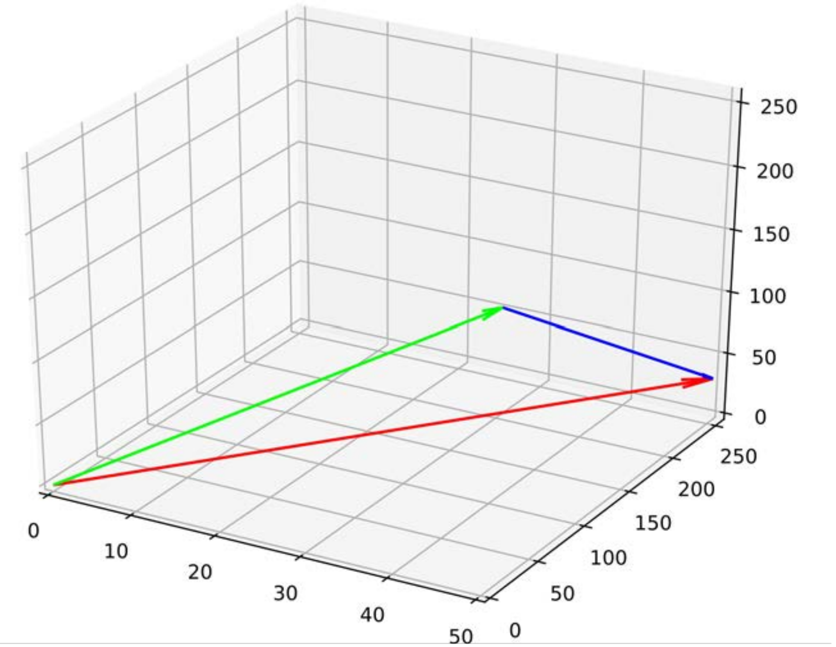
$\boldsymbol{X_0}$ Image I(x, y) (9)

- Relatively expensive sqrt computation, however:

$$\mathrm{argmin}_{\boldsymbol{X}_i}\|\boldsymbol{X}_i - \boldsymbol{T}\| = \mathrm{argmin}_{\boldsymbol{X}_i}\|\boldsymbol{X}_i - \boldsymbol{T}\|^{\boldsymbol{2}}$$

- distance is minimized ⇔squared distance is minimized

$$\mathrm{argmin}_{\boldsymbol{X}_i}(\boldsymbol{X}_i - \boldsymbol{T})^T(\boldsymbol{X}_i - \boldsymbol{T})$$



2D Template Matching

- Let's look at a 2D example

row 0 →

50	255	90
80	200	100
150	90	30

↑ column 0

Template $\boldsymbol{T}$ (3×3)

$$\boldsymbol{T} = \begin{bmatrix} 50 \\ 255 \\ 90 \\ 80 \\ 200 \\ 100 \\ 150 \\ 90 \\ 30 \end{bmatrix}$$

row r →

50	255	30	80	30	100	50	200	250
80	200	100	50	60	30	30	60	30
150	90	30	80	90	100	250	100	240

↑ column c

$\boldsymbol{X}_4$

Image 2D Array $\mathrm{I}(x, y)$
(9×3)
i.e. 7 3×3 patches

$$\boldsymbol{X}_4 = \begin{bmatrix} 80 \\ 30 \\ 100 \\ 50 \\ 60 \\ 30 \\ 80 \\ 90 \\ 100 \end{bmatrix}$$

- Now our template and image patches are 3×3 patches or 9D vectors

2D Template Matching

- Consider the template (or any patch) to have it's origin (0,0) in the patch centre
- Instead of explicitly remapping our template/patches into 1D vectors, we can use the write an expression based on the 2D arrays
- For each patch at location (r, c) in image I, we calculate the 2D sum:

$$\mathrm{dist}(r,c) = \sqrt{\sum_{a=-N}^{N} \sum_{b=-N}^{N} \left(I(r+a, c+b) - T(a,b)\right)^2}$$

where N is the "radius" of a patch, i.e. 1 for our 3×3 template with indices in the range [-1,1]

50	255	30	80	30	100	50	200	250
80	200	100	50	60	30	30	60	30
150	90	30	80	90	100	250	100	240

row r

column c

$\boldsymbol{X_4}$

Image 2D Array $\mathrm{I}(x,y)$
(9×3)
i.e. 7 3×3 patches

50	255	90
80	200	100
150	90	30

row 0

column 0

Template $\boldsymbol{T}$
(3×3)

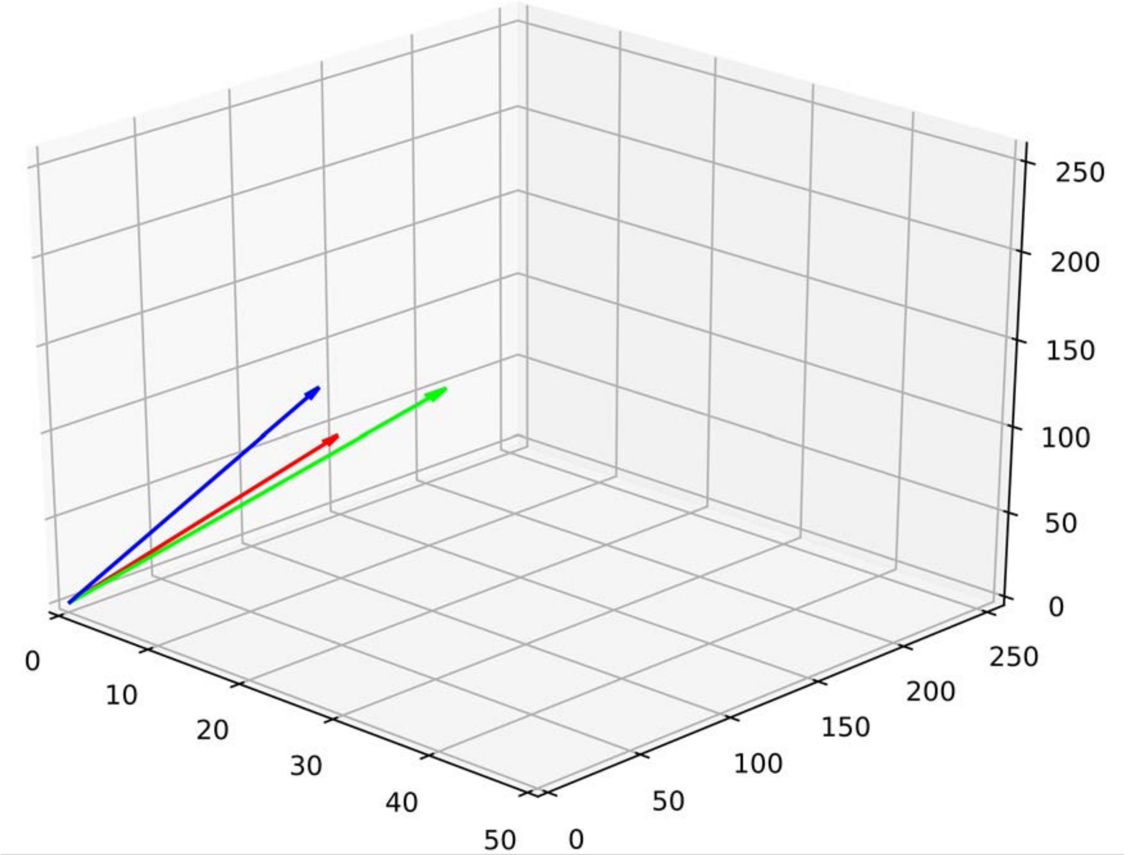
Basic Template Matching Algorithm

$$\mathrm{dist}(r,c) = \sqrt{\sum_{a=-N}^{N} \sum_{b=-N}^{N} \big(I(r+a, c+b) - T(a,b)\big)^2}$$

- Define an "output" image of size equal to the "input" image I
- Compute $\mathrm{dist}(r,c)$ for every pixel location r, c in image I where the computation is possible
  - i.e. not on the "border" pixels where the template does not "fit"
- Search the image for the location of the lowest distance value – this location is the closest match

Distance as Similarity

- Let’s think about what our similarity metric means…
- Which of the vectors is closest to the red vector?
- Blue – and distance will tell us this

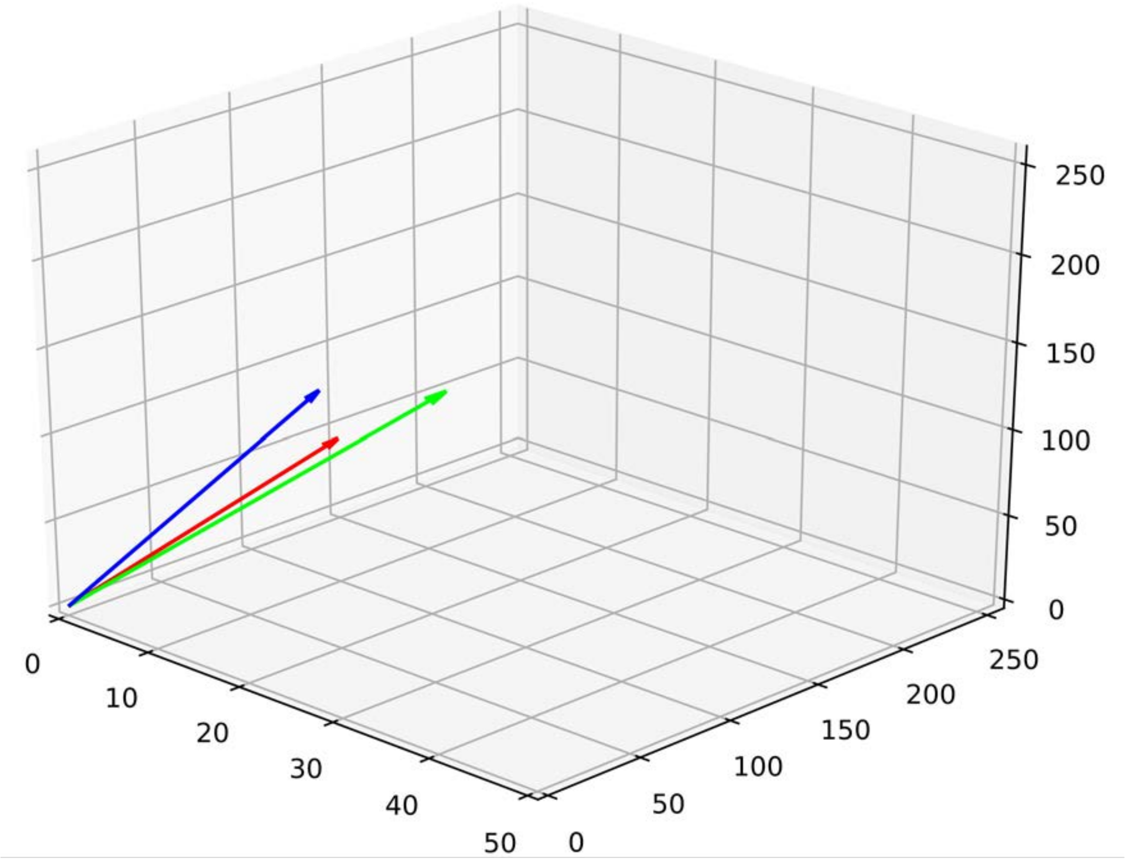


- **But** what about if the vectors are image patches…

Scaled Image Example

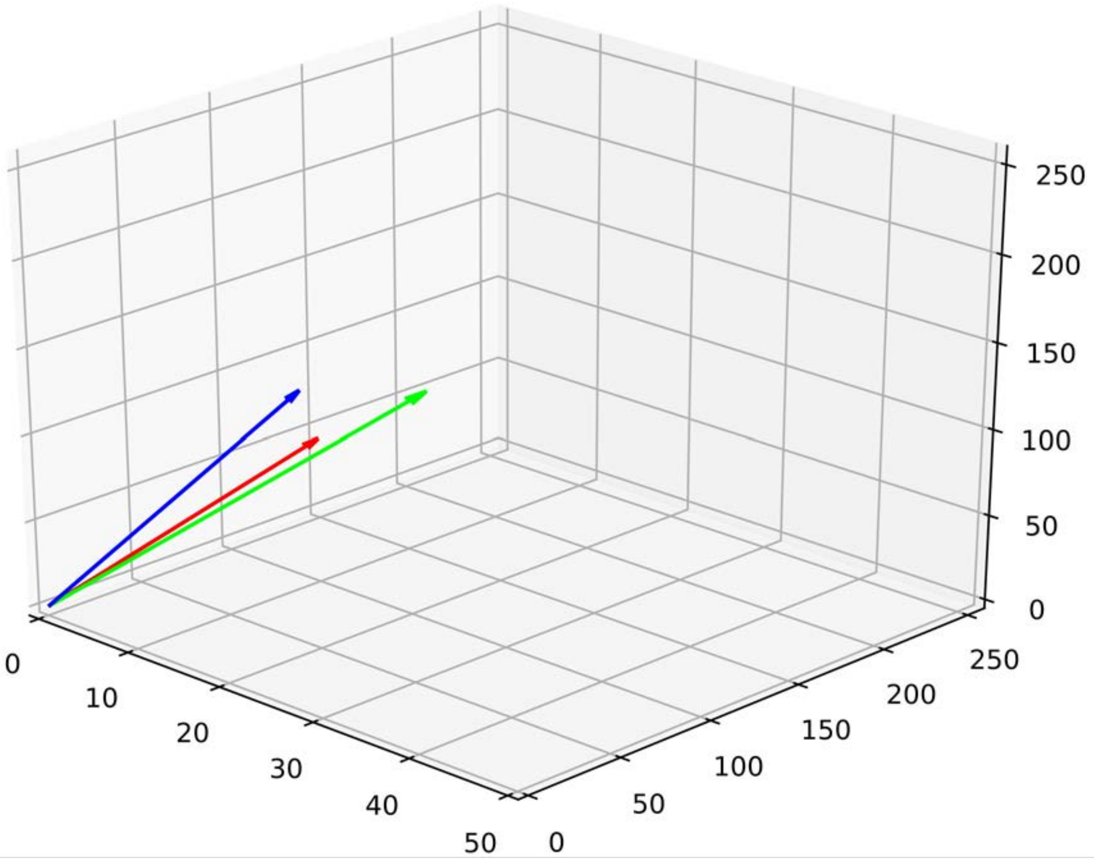
Distance as Similarity

- If vectors are image patches, the green vector is a **scaled** version of the red vector (i.e. brighter image), with some noise!
- Blue is different in some other ways, that are probably more perceptible to us as Humans!



Distance as Similarity

- We would tend to see images with different scaling (brightness) as very similar, at least compared to other changes
- Our distance-based similarity metric cannot distinguish between patches that are just scaled versions of the template *T*, and patches that differ in other ways!



Topic 6: 2D Images and Curves

- Images as Vectors
- Template Matching
- **Cross-Correlation**
- Template Matching using Cross-Correlation
- Dimensionality Reduction
- Principle Component Analysis (PCA)
- Understanding PCA
- Case Study: EigenFaces

Similarity Metric #2: Cross-Correlation

- Goal: Find the image patch $x_i \in I$ that is **most similar** to given template $\boldsymbol{T}$
- Let's define a new similarity metric:

$$\mathrm{CC}\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = \boldsymbol{X}_i^T \cdot \boldsymbol{T}$$

i.e. the dot-product of the vectors $\boldsymbol{X}_i, \boldsymbol{T}$

- This is called the **cross-correlation,** or more intuitively the “sliding dot-product”
- Why is this nicer than distance?

20	300	21

Template $\boldsymbol{T}$

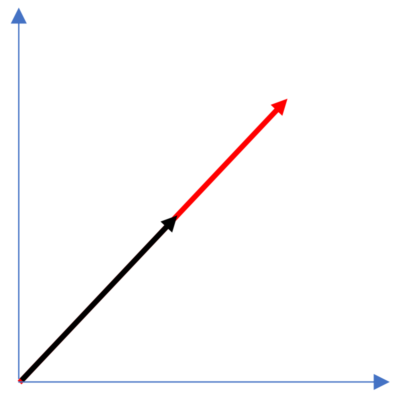
50	255	30	50	3	122	176	220	160

$\boldsymbol{X}_0$ Image $\mathrm{I}(x, y)$
(9)

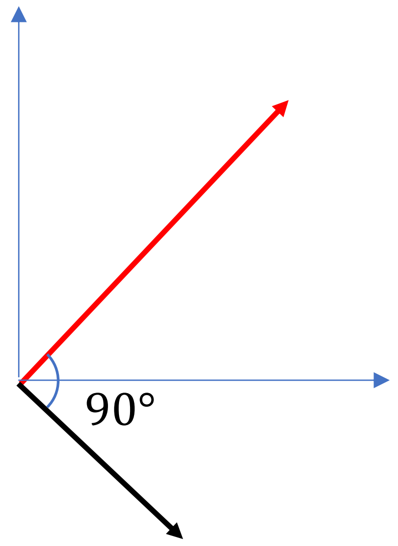
Cross-Correlation as a Similarity Metric

- Recall the dot product is also defined:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\|\|\boldsymbol{b}\| \cos\theta$$



- Depends on the angle between the vectors
  - If θ is small, dot product is large
  - Maximized when $\boldsymbol{a}, \boldsymbol{b}$ are in the same direction (i.e. $\theta = 0°$)
  - Zero when $\boldsymbol{a}, \boldsymbol{b}$ are orthogonal, i.e. $\theta = 90°$
- Also depends on the length of the vectors $\boldsymbol{a}, \boldsymbol{b}$

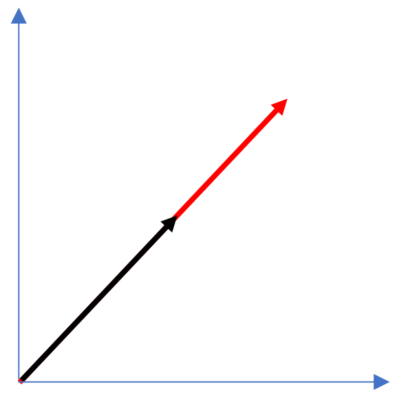


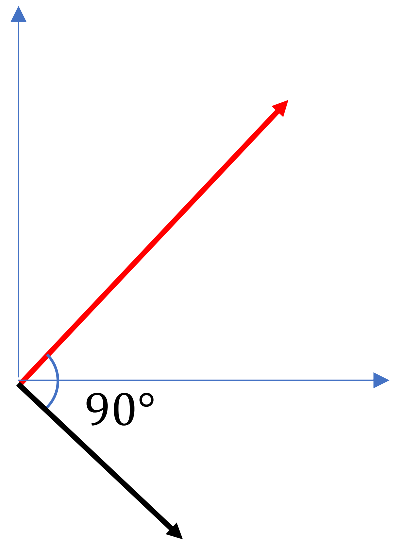
Similarity Metric #3: Normalized Cross-Correlation

$$\mathrm{CC}\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = \boldsymbol{X}_i^T \cdot \boldsymbol{T} = \left\|\boldsymbol{X}_i^T\right\| \|\boldsymbol{T}\| \cos\theta$$

- It is somewhat intuitive that we want image patch vectors with similar directions to be considered similar
- But this measure clearly biases towards vectors with larger lengths – this doesn't make much sense
- Instead, let's normalize the result so it is independent of the vector magnitudes…

$$\mathrm{NCC}\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = \frac{\boldsymbol{X}_i^T \cdot \boldsymbol{T}}{\left\|\boldsymbol{X}_i^T\right\| \|\boldsymbol{T}\|}$$

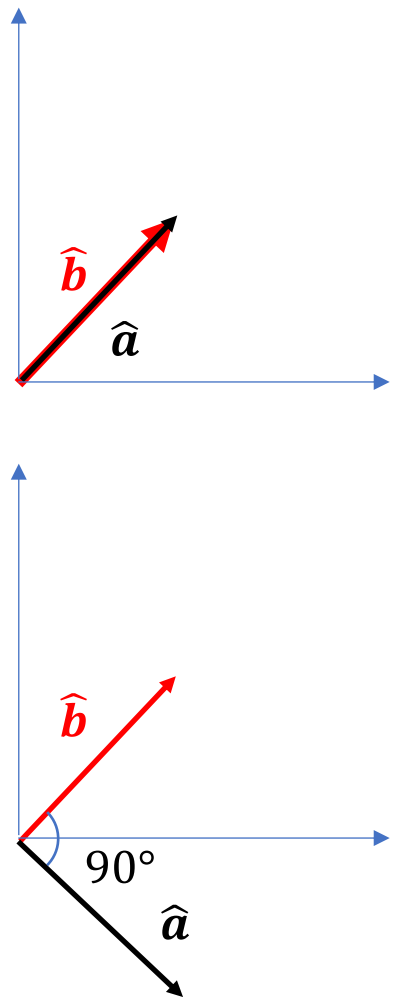




Normalized Cross-Correlation

$$\mathrm{NCC}\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = \frac{\boldsymbol{X}_i^T \cdot \boldsymbol{T}}{\left\|\boldsymbol{X}_i^T\right\| \|\boldsymbol{T}\|} = \cos\theta$$

- This is actually just the cosine of the angle between the vectors! (or dot product of unit vectors $\widehat{\boldsymbol{X}_i^T}, \widehat{\boldsymbol{T}}$)
- Properties:
  - Independent of norm of image patches (length of vector)
  - $= 1.0$ (max) when the intensities of $\boldsymbol{X}_i^T, \boldsymbol{T}$ are identical (to a scale factor)
  - $= 0.0$ (min) when $\boldsymbol{X}_i^T, \boldsymbol{T}$ are orthogonal (most dissimilar)



Topic 6: 2D Images and Curves

- Images as Vectors
- Template Matching
- Cross-Correlation
- **Template Matching using Cross-Correlation**
- Dimensionality Reduction
- Principle Component Analysis (PCA)
- Understanding PCA
- Case Study: EigenFaces

2D Template Matching Using Cross-Correlation

- For each patch at location (r, c) in image I, we calculate the 2D sum:

$$\mathrm{CC}(r,c) = \sum_{a=-N}^{N} \sum_{b=-N}^{N} I(r+a, c+b)\, T(a,b)$$

where N is the "radius" of a patch as before.

	50	255	30	80	30	100	50	200	250
row r	80	200	100	50	60	30	30	60	30
	150	90	30	80	90	100	250	100	240

column c $\boldsymbol{X_4}$

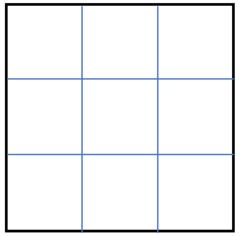
Image 2D Array $\mathrm{I}(x,y)$
(9×3)
i.e. 7 3×3 patches

	50	255	90
row 0	80	200	100
	150	90	30

column 0

Template $\boldsymbol{T}$
(3×3)

Template Matching: Computational Issues



Template $\boldsymbol{T}$
(M pixels)

$$\boldsymbol{T} = \begin{bmatrix} t_0 \\ \vdots \\ t_M \end{bmatrix}$$

Image 2D Array $\mathrm{I}(x, y)$
(N pixels)

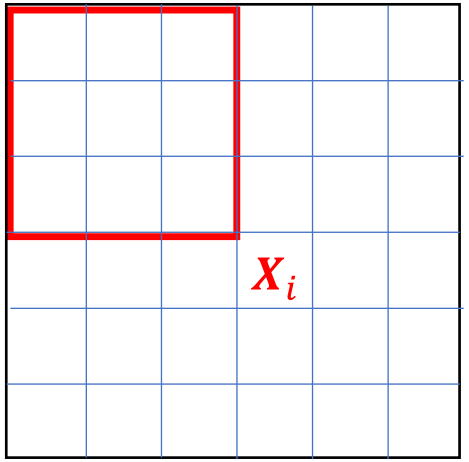


Image Patch $\boldsymbol{X}_\mathrm{i}(x, y)$
(M pixels)

$$\boldsymbol{X}_i = \begin{bmatrix} x_0 \\ \\ x_M \end{bmatrix}$$

- Assume a template with M pixels, and an image with N pixels
  - For example, if our image is 1000×1000, $N = 10^6$
  - If our template is 32×32, $M = 32^2 = 1024$
- For **each patch**, the CC metric requires M multiplications, $M - 1$ additions
- Total $O(NM)$ operations for entire image!

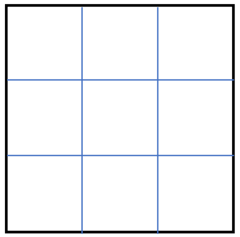
Similarity Metrics:

$$\mathrm{dist}^2\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = (\boldsymbol{X}_i - \boldsymbol{T})^T(\boldsymbol{X}_i - \boldsymbol{T})$$

$$\mathrm{CC}\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = \boldsymbol{X}_i^T \cdot \boldsymbol{T}$$

$$\mathrm{NCC}\left(\boldsymbol{X}_i^T, \boldsymbol{T}\right) = \frac{\boldsymbol{X}_i^T \cdot \boldsymbol{T}}{\|\boldsymbol{X}_i^T\|\|\boldsymbol{T}\|}$$

Template Matching: Computational Issues



Template $\boldsymbol{T}$
(M pixels)

$$\boldsymbol{T} = \begin{bmatrix} t_0 \\ \vdots \\ t_M \end{bmatrix}$$

Image 2D Array $\mathrm{I}(x, y)$
(N pixels)

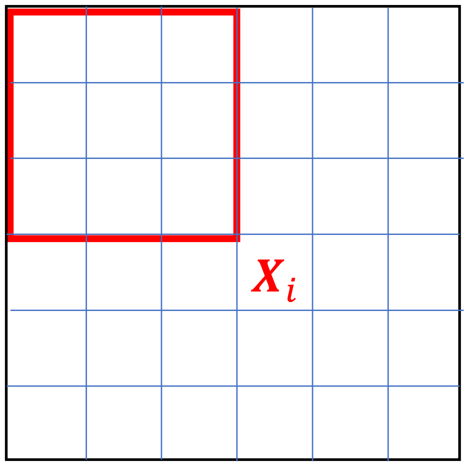


Image Patch $\boldsymbol{X}_{\mathrm{i}}(x, y)$
(M pixels)

$$\boldsymbol{X}_i = \begin{bmatrix} x_0 \\ \\ x_M \end{bmatrix}$$

- Total $O(NM)$ operations for entire image, where N and M are very large
- Clearly template matching is very expensive!
- What if we could represent patches $\boldsymbol{X}_i^T, \boldsymbol{T}$ with only $d \ll M$ dimensions
- Would have $O(dM)$

Dimensionality Reduction

Topic 6: 2D Images and Curves

- Images as Vectors
- Template Matching
- Cross-Correlation
- Template Matching using Cross-Correlation
- **Dimensionality Reduction**
- Principle Component Analysis (PCA)
- Understanding PCA
- Case Study: EigenFaces

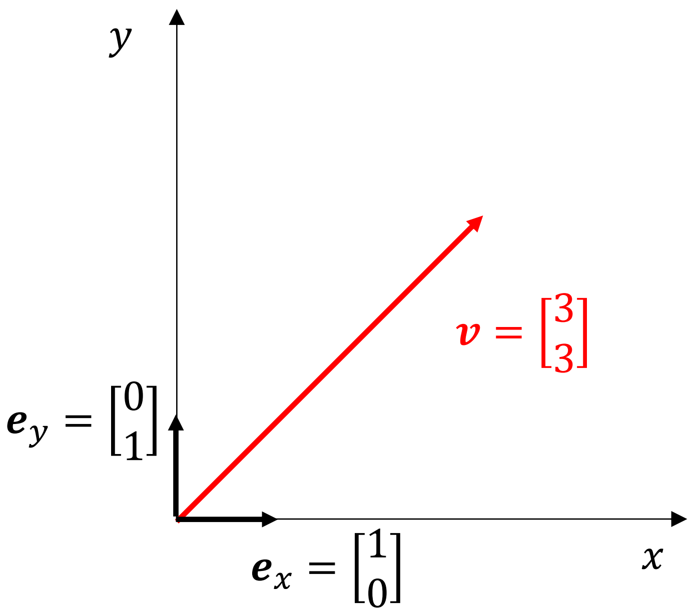
Math Refresher: Basis Vectors

- Vectors are expressed relative to **basis**
- Typically this is the standard basis, i.e. for the Euclidean 2D space, the **basis vectors** are:

$$\boldsymbol{e}_x = (0, 1) \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \boldsymbol{e}_y = (1,0) \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Any vector we have in this space is uniquely represented as a linear combination of the basis vectors, e.g.:

$$\boldsymbol{v} = (3, 3) = 3\boldsymbol{e}_x + 3\boldsymbol{e}_y$$

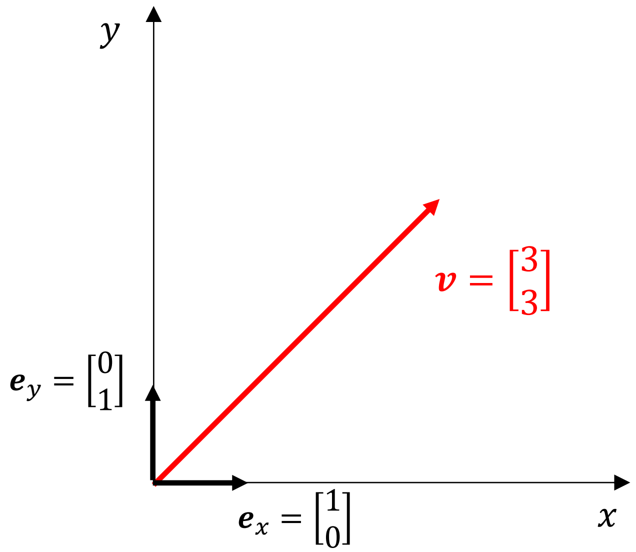


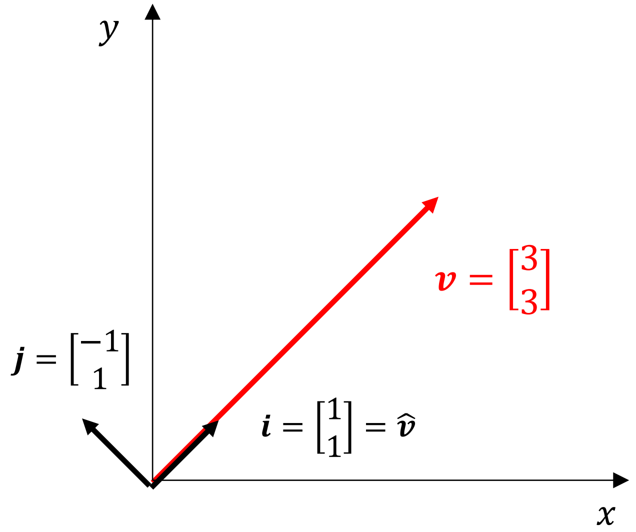
Math Refresher: Change of Basis

- We can use a non-standard basis to represent any vector
- For example, perhaps we want to represent this vector with a new basis B:

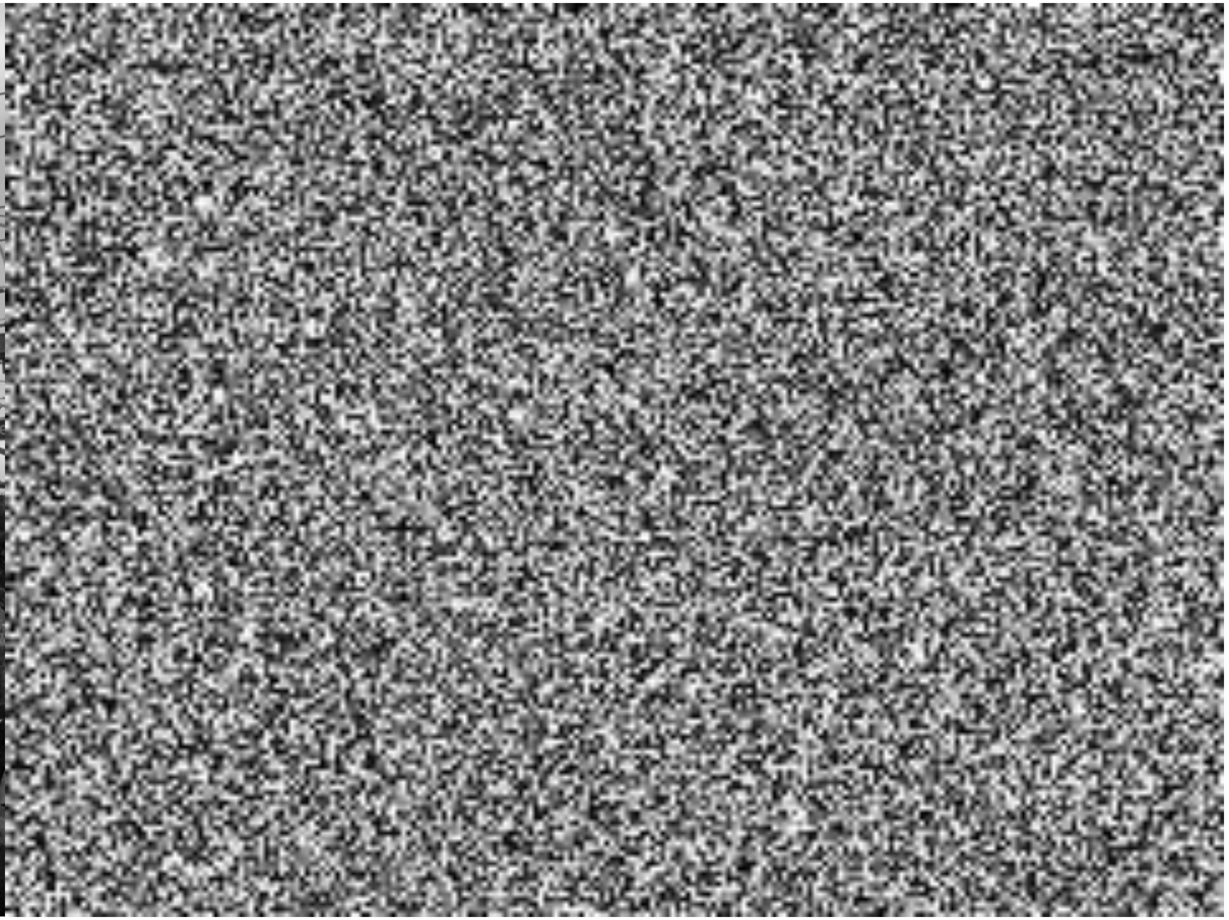
$$\boldsymbol{i}_B = (1,1), \boldsymbol{j}_B = (-1,1)$$

- Notice that in this new basis one of our basis vectors is the unit vector $\hat{\boldsymbol{v}}$
- Under our new basis, $\boldsymbol{v}_B = 3\boldsymbol{i}_B \equiv (3, 0)$



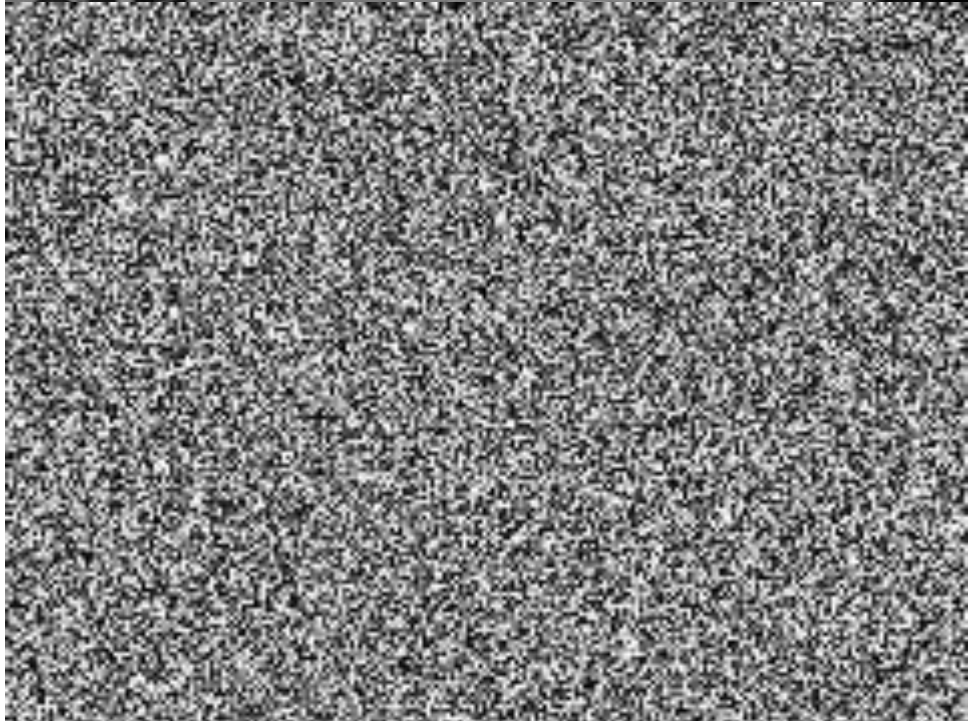


Natural Images are not Random



Natural Images

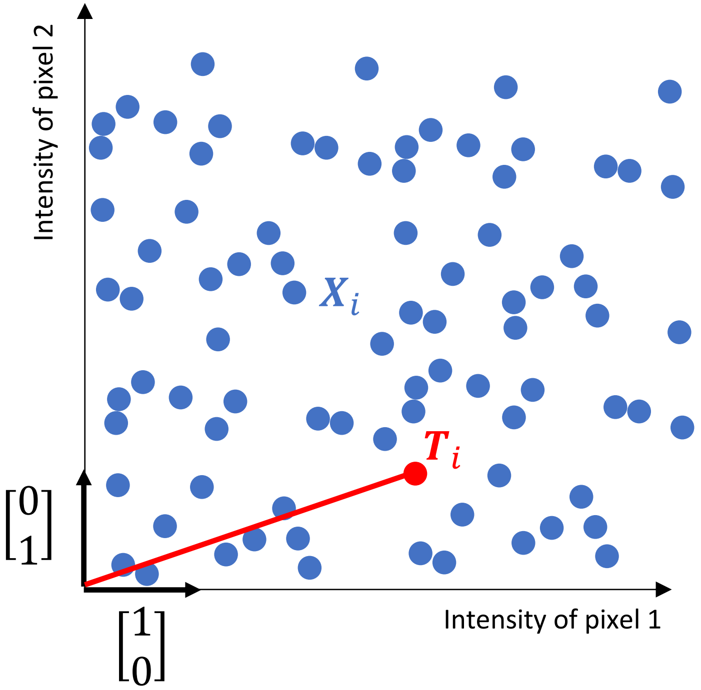
- We would not expect to see the bottom image out of our camera! (white noise)
- However, both of these are valid vectors in the same N-dimensional image space!
- Natural images have **structure**
- Even if we considered all possible natural images, they would occupy only a fraction of the full N-dimensional space
- How can we take advantage of this?



Linear Dimensionality Reduction: Intuition

Case A: *pixel intensities are* ***unrelated***

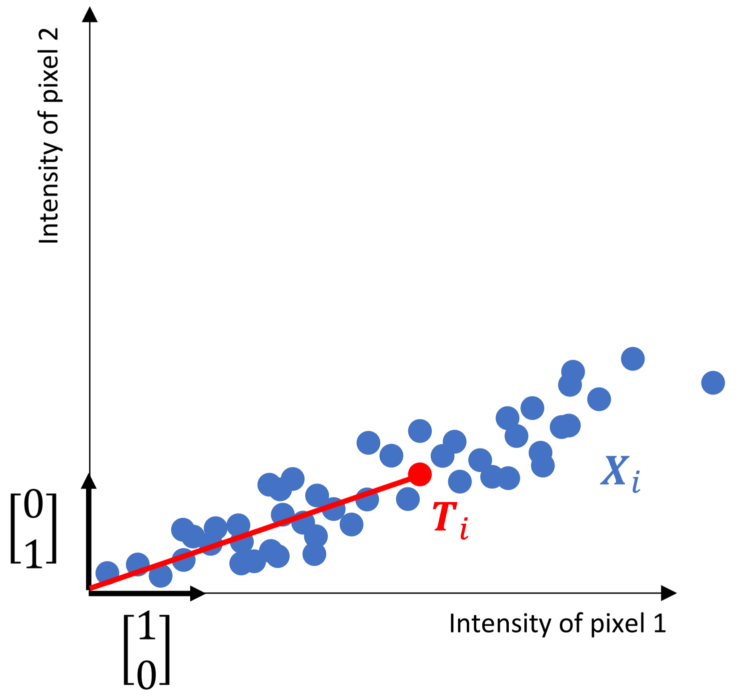
- What would we expect our **space** of image patches to look like?
- For simplicity, assume our patches/templates are 2-pixels long!
- Image patch vectors should look random, uncorrelated, with no discernable relationship between pixels



Linear Dimensionality Reduction: Intuition

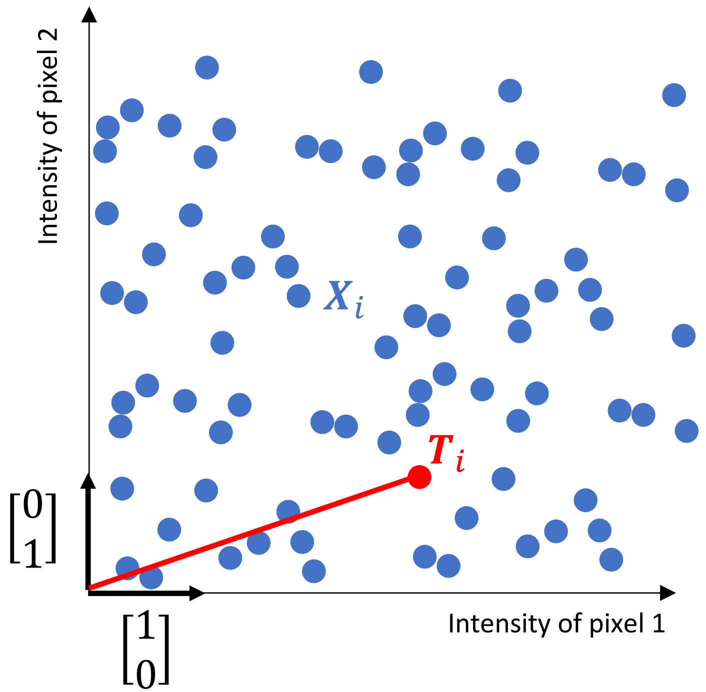
Case B: *pixel intensities are **related***

- What would we expect our space of image patches to look like?
- Image patch vectors have a trend, are correlated, with relationships between pixels

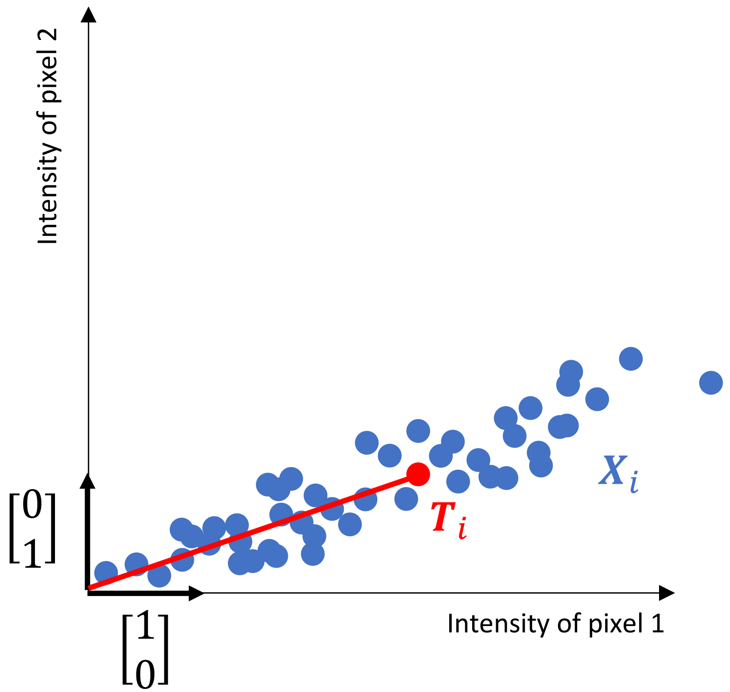


Linear Dimensionality Reduction: Intuition

Case A: *pixel intensities are* ***unrelated***

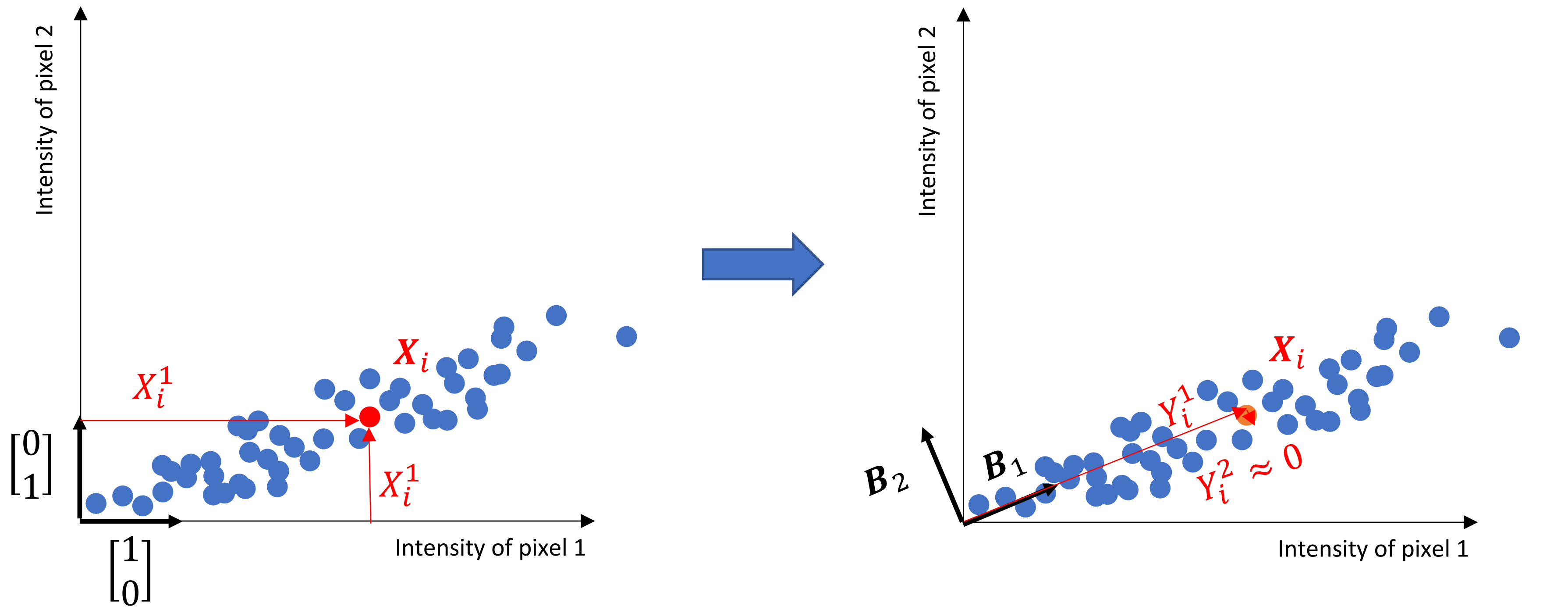


Case B: *pixel intensities are* ***related***



Linear Dimensionality Reduction: Intuition

- What happens if we change basis?

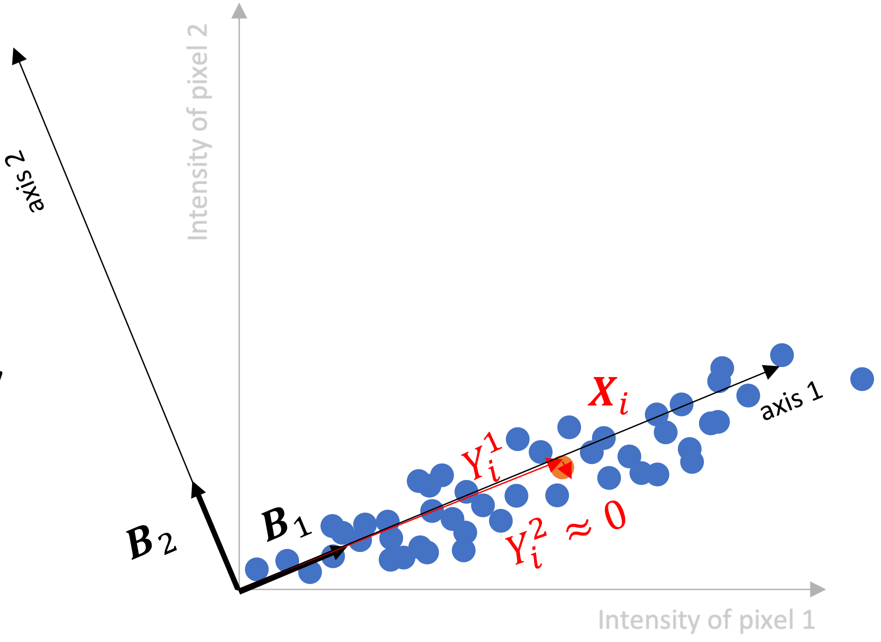


$$\boldsymbol{X}_i = \begin{bmatrix} X_i^1 \\ X_i^2 \end{bmatrix} = X_i^1 \boldsymbol{e}_x + X_i^2 \boldsymbol{e}_y$$

$$\boldsymbol{X}_i = \begin{bmatrix} Y_i^1 \\ Y_i^2 \end{bmatrix} = Y_i^1 \boldsymbol{B}_1 + \overset{\approx 0}{Y_i^2} \boldsymbol{B}_2 \approx Y_i^1 \boldsymbol{B}_1$$

Linear Dimensionality Reduction: Intuition

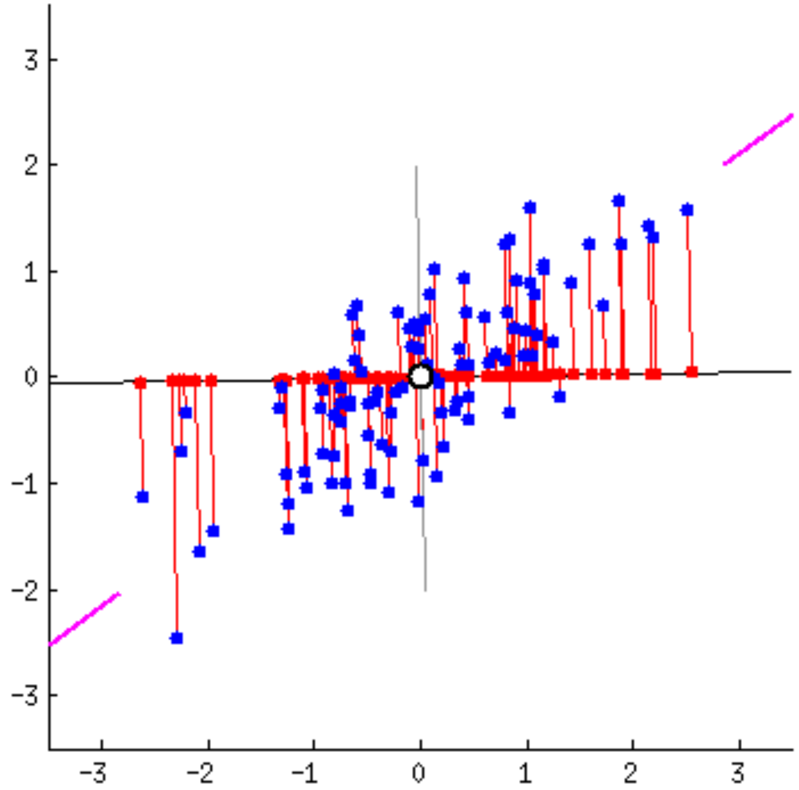
- Idea: When pixel intensities are related, we can express a patch in terms of basis vectors where **only a few of the coordinates** are significant (i.e. not close to 0)
- This is a nutshell is linear dimensionality reduction: **remove unneeded dimensionality**



- This really depends on the **basis** we choose!

$$\boldsymbol{X}_i = \begin{bmatrix} Y_i^1 \\ Y_i^2 \end{bmatrix} = Y_i^1 \boldsymbol{B}_1 + Y_i^2 \boldsymbol{B}_2 \approx Y_i^1 \boldsymbol{B}_1$$

How Vector Components Change with Basis



https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

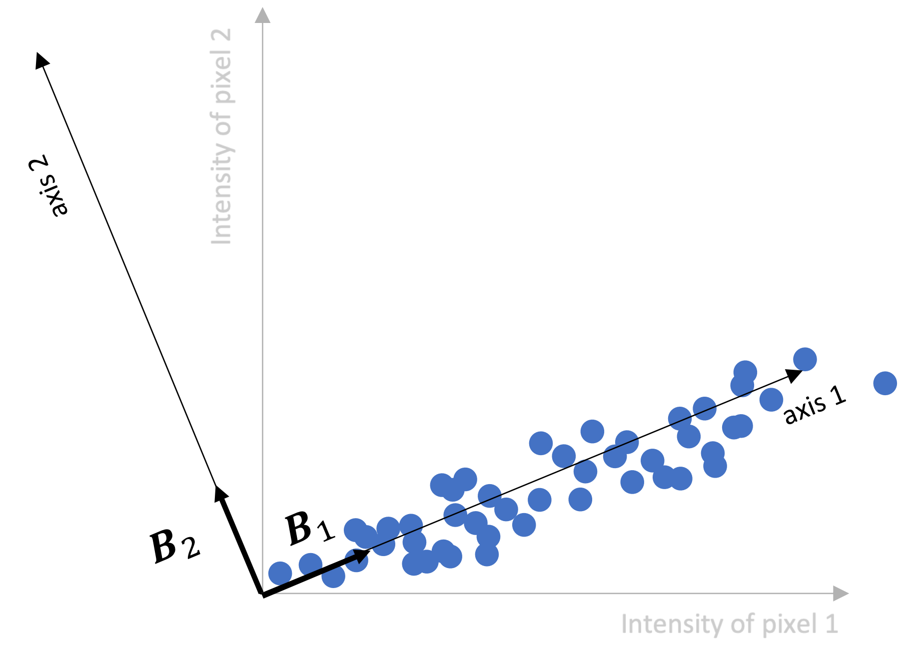
Intermission

Topic 6: 2D Images and Curves

- Images as Vectors
- Template Matching
- Cross-Correlation
- Template Matching using Cross-Correlation
- Dimensionality Reduction
- **Principle Component Analysis (PCA)**
- Understanding PCA
- Case Study: EigenFaces

Principle Components

- What we want to find are the **principle components** of the data, i.e. the directions in which the data shows the most variation
- First we need to know how to change basis!



Changing Basis: Matrix Notation

Single Patch:

(standard basis)

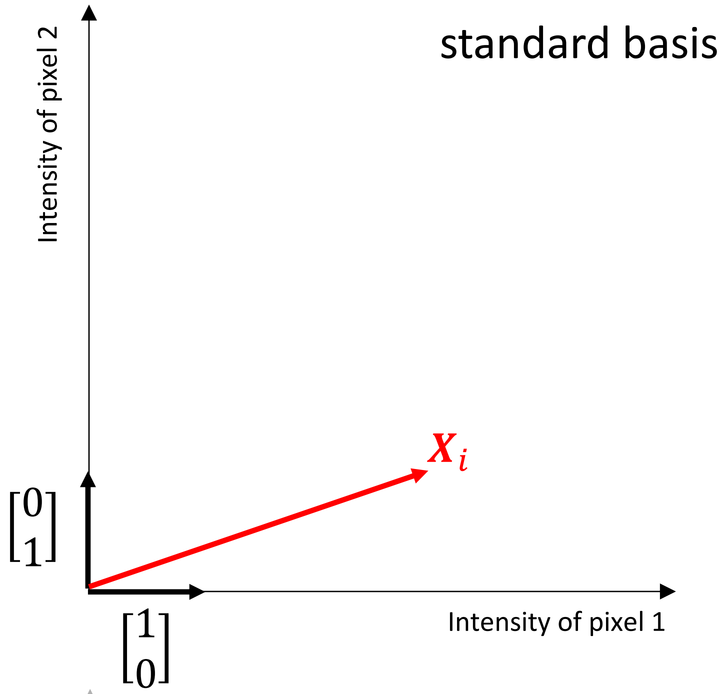
$$\boldsymbol{X}_i = \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{\substack{\text{basis matrix} \\ \text{(columns are basis vectors)}}} \underbrace{\begin{bmatrix} X_i^1 \\ X_i^2 \end{bmatrix}}_{\text{coordinate vector}}$$

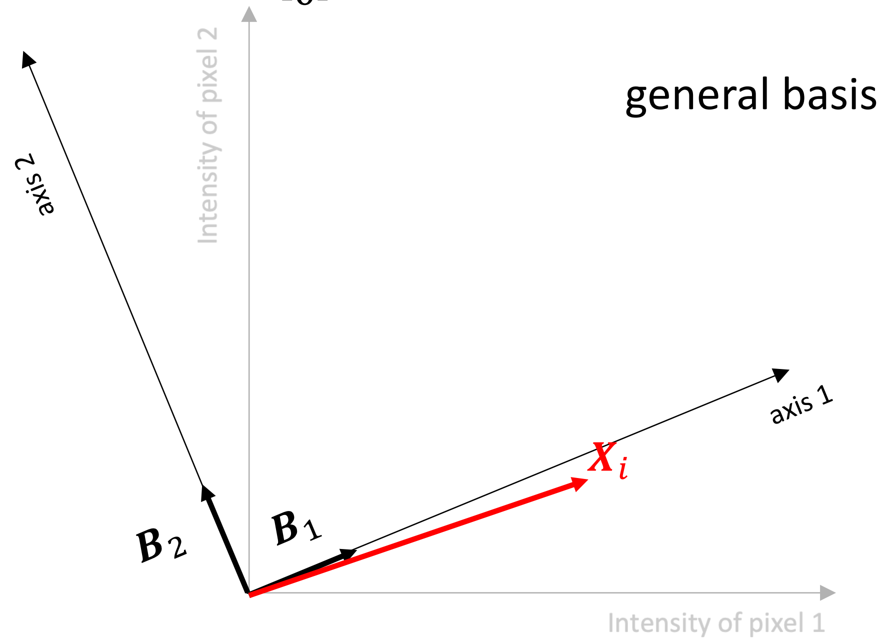
(general basis)

$$\boldsymbol{X}_i = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \end{bmatrix} \begin{bmatrix} Y_i^1 \\ Y_i^2 \end{bmatrix}$$

N 2-dim Patches:

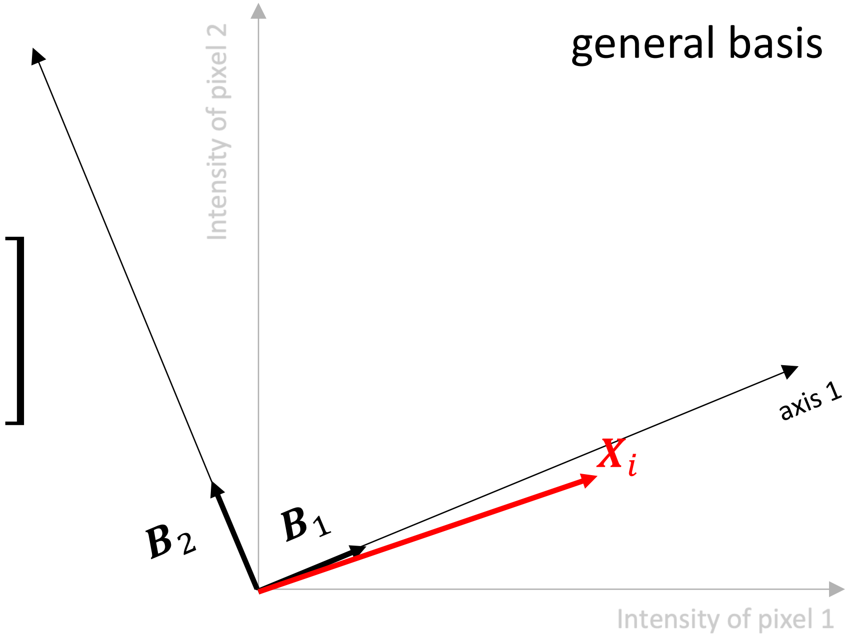
$$\boldsymbol{X}_i = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \end{bmatrix} \begin{bmatrix} Y_1^1 & Y_2^1 & \cdots & Y_N^1 \\ Y_1^2 & Y_2^2 & \cdots & Y_N^2 \end{bmatrix}$$





Changing Basis: Matrix Notation

N 2-dim Patches: $\boldsymbol{X}_i = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \end{bmatrix} \begin{bmatrix} Y_1^1 & Y_2^1 & \cdots & Y_N^1 \\ Y_1^2 & Y_2^2 & & Y_N^2 \end{bmatrix}$



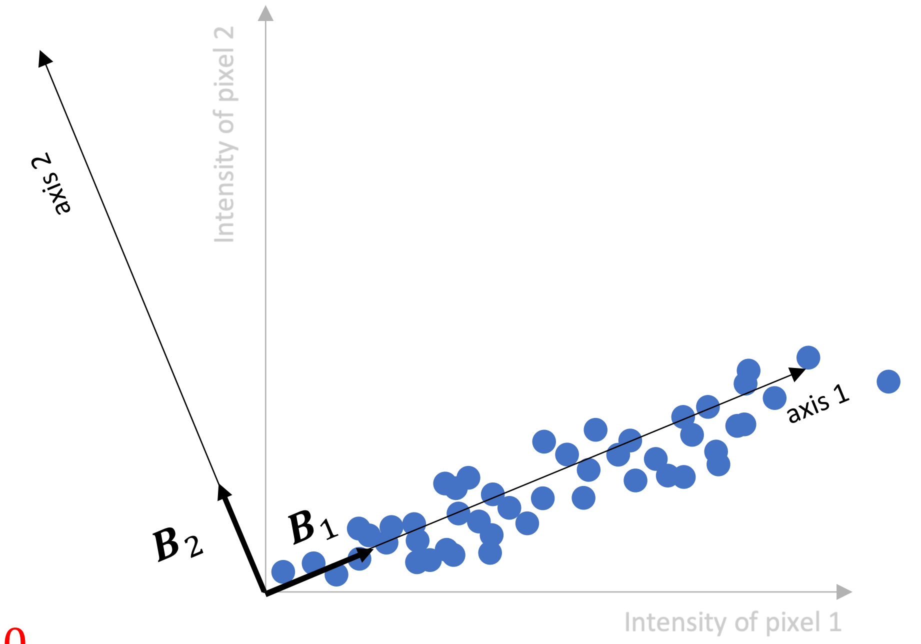
N M-dim Patches:

$$[\boldsymbol{X}_1 \quad \boldsymbol{X}_2 \cdots \quad \boldsymbol{X}_N] = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \cdots \boldsymbol{B}_M \end{bmatrix} \begin{bmatrix} Y_1^1 & Y_2^1 & & Y_N^1 \\ Y_1^2 & Y_2^2 & \cdots & Y_N^2 \\ Y_1^M & Y_2^M & & Y_N^M \end{bmatrix}$$

Changing Basis: Matrix Notation

- Want to choose $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_M$ such that $Y_1^j \approx 0$ for $d < j \leq M$, i.e.:

$$[\boldsymbol{X}_1 \quad \boldsymbol{X}_2 \cdots \quad \boldsymbol{X}_N] = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \cdots \boldsymbol{B}_M \end{bmatrix} \begin{bmatrix} Y_1^1 & Y_2^1 & & Y_N^1 \\ \vdots & \vdots & & \vdots \\ Y_1^d & Y_2^d & \cdots & Y_N^d \\ \color{red}{Y_1^{d+1}} & \color{red}{Y_2^{d+1}} & & \color{red}{Y_N^{d+1}} \\ \color{red}{\vdots} & \color{red}{\vdots} & & \color{red}{\vdots} \\ \color{red}{Y_1^M} & \color{red}{Y_2^M} & & \color{red}{Y_N^M} \end{bmatrix} \begin{matrix} \\ \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \color{red}{\text{all} \approx 0} \end{matrix}$$



Changing Basis: Matrix Notation

$$[\boldsymbol{X}_1 \quad \boldsymbol{X}_2 \cdots \quad \boldsymbol{X}_N] = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \cdots \boldsymbol{B}_d \end{bmatrix} \begin{bmatrix} Y_1^1 & Y_2^1 & & Y_N^1 \\ \vdots & \vdots & & \vdots \\ Y_1^d & Y_2^d & \cdots & Y_N^d \\ \cancel{Y_1^{d+1}} & \cancel{Y_2^{d+1}} & & \cancel{Y_N^{d+1}} \\ \vdots & \vdots & & \vdots \\ \cancel{Y_1^M} & \cancel{Y_2^M} & & \cancel{Y_N^M} \end{bmatrix}$$

- Assume we find such a basis $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_M$
- We can approximate the the patches using only the first d components of the patch vectors
- We have a d-dimensional approximation!

Principle Component Analysis (PCA) Algorithm

Given N image patches of M dimensions:

1) Calculate mean of image patch vectors

$$\overline{\boldsymbol{X}} = \frac{1}{N} \sum \boldsymbol{X}_i$$

2) Subtract the mean from all patches (centre)

$$\boldsymbol{Z}_i = \boldsymbol{X}_i - \overline{\boldsymbol{X}}$$

3) Create an $M \times N$ matrix of all centred patch vectors (arranged as columns of matrix)

$$Z = [\boldsymbol{Z}_1 \quad \boldsymbol{Z}_2 \cdots \quad \boldsymbol{Z}_N]$$

4) Find ***eigenvectors*** $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_d$ corresponding to the d (where $d \ll M$) largest ***eigenvalues*** $\lambda_1, \ldots, \lambda_d$ of the **covariance matrix**

$$\Sigma = ZZ^T$$

Topic 6: 2D Images and Curves

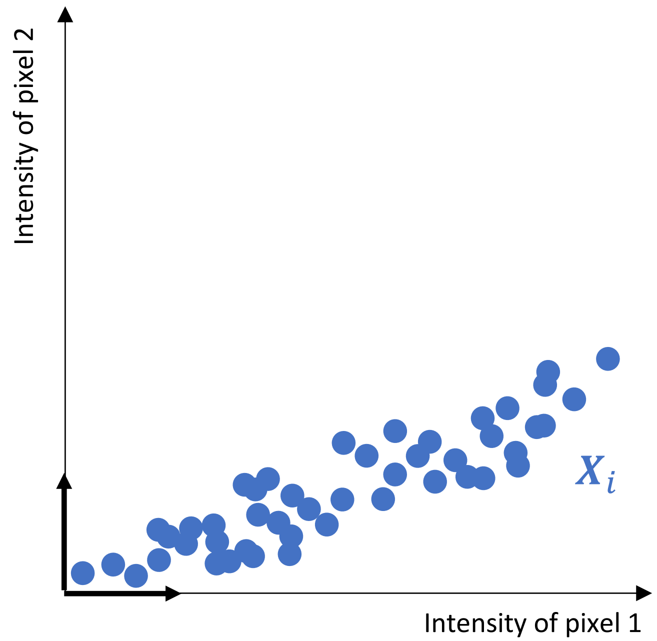
- Images as Vectors
- Template Matching
- Cross-Correlation
- Template Matching using Cross-Correlation
- Dimensionality Reduction
- Principle Component Analysis (PCA)
- **Understanding PCA**
- Case Study: EigenFaces

Understanding PCA

- This is the algorithm (that you should know), and it’s immensely useful – possibly one of the most useful things you can learn in this course
- However, we haven’t explained why it works! Or for that matter what eigenvectors/eigenvalues are…
- Here we will attempt to gain an understanding what PCA is doing, and why it works
- You should take away at least the following: what is the covariance matrix, and the SVD of the covariance gives us the eigenvectors/values

Principle Component Analysis (PCA): Intuition

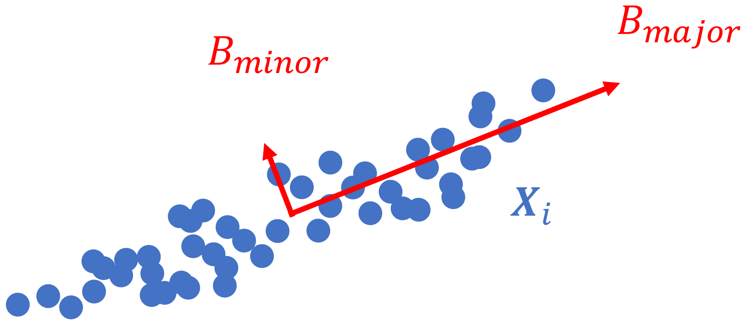
- This is all great! We understand we want to find a basis of principle components
- But how do we find this basis?
- Let's look at an example, here (again) are our 2-dimensional image patches



Principle Component Analysis (PCA): Intuition

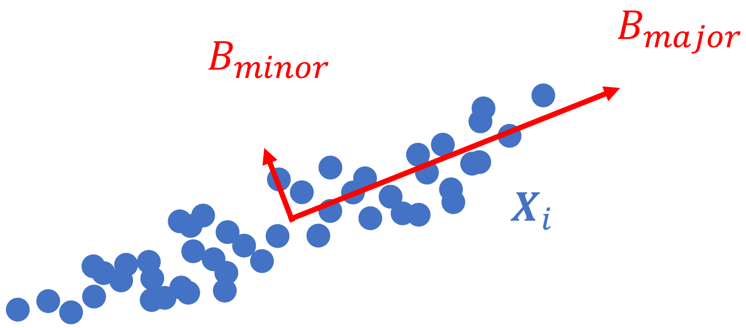
- Let's look at an example, here (again) are our 2-dimensional image patches

- We would like to find two orthogonal vectors:
  - major direction of the largest data variance
  - minor direction of least variance



Principle Component Analysis (PCA): Intuition

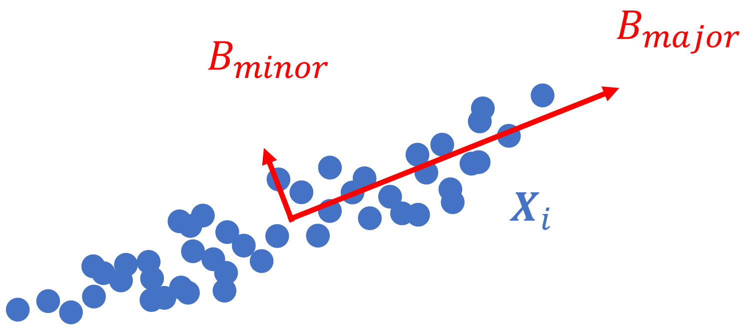
- Let’s put our computer vision hat back on for a minute…
- Can we find a shape to contain this data that would tell us the major and minor axis of variation?



- Hint: not a line – line only gives us one direction!

Principle Component Analysis (PCA): Intuition

- Let's put our computer vision hat back on for a minute…
- Can we find a shape to contain this data that would tell us the major and minor axis of variation?

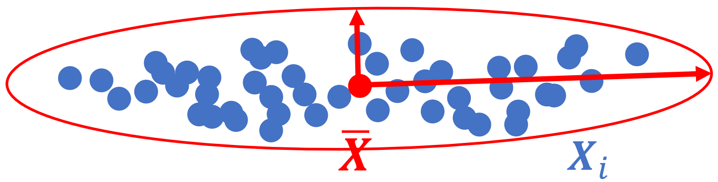


- Hint: not a line – line only gives us one direction!
- Hint2: what shape has a major and minor axis?

PCA as Ellipse Fitting

- This is similar to modelling our data's variance using an ellipse!
- Equation of an ellipse:

$$\frac{(x-c_1)^2}{a^2} + \frac{(y-c_2)^2}{b^2} = 1$$



- The centre of the ellipse is simply the mean of data:

$$\bar{\boldsymbol{X}} = \frac{1}{N}\sum \boldsymbol{X}_i$$

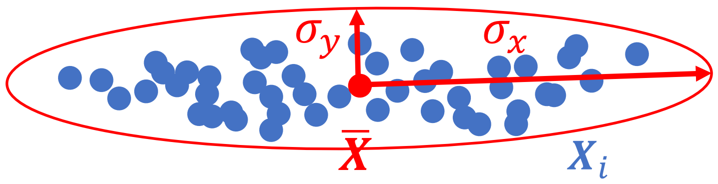
- We can subtract this mean from our data, giving us an ellipse centred on the origin:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

PCA as Ellipse Fitting

- What are the lengths of our major and minor axis?

$$\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} = 1$$



- "Spread" i.e. standard deviation of our data in the x, y axes:

$$\sigma_x, \sigma_y$$

- Recall, the sample variance of a dataset X (where $\bar{X}$ is the mean):

$$\sigma^2(X) = \frac{1}{N} \sum (X_i - \bar{X})^2$$

PCA as Ellipse Fitting

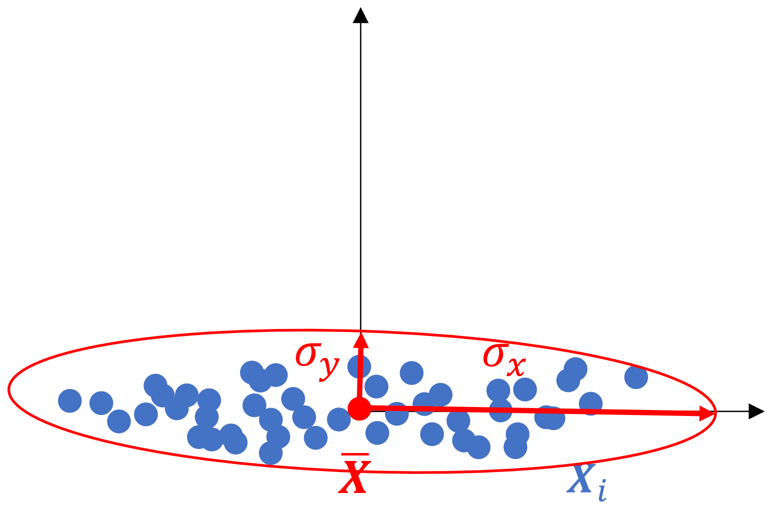
- What about a non-axis aligned eclipse?

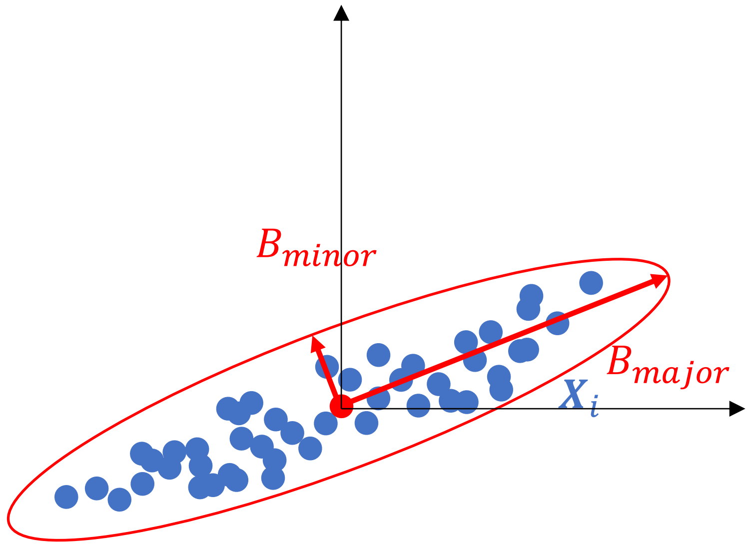
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

- In our equation of the ellipse, a, b are the x and y components of major/minor axis
- Variance is also only defined in terms of x and y components!

$$\sigma_X^2(X^x) = \frac{1}{N}\sum(X_i^x - \bar{X}^x)^2$$

- In 2D, we must look at more general form of variance: **covariance**





Covariance

- We define covariance:

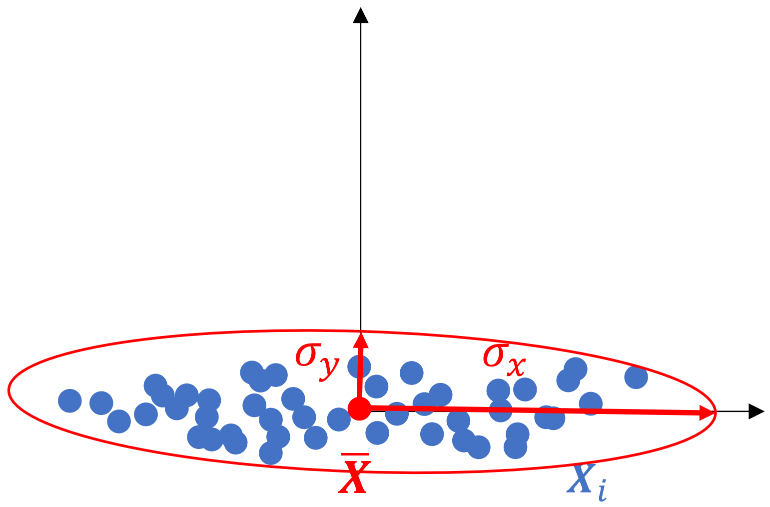
$$\mathrm{cov}(X,Y) = \frac{1}{N}\sum (X_i - \bar{X}^X)(Y_i - \bar{X}^Y)$$

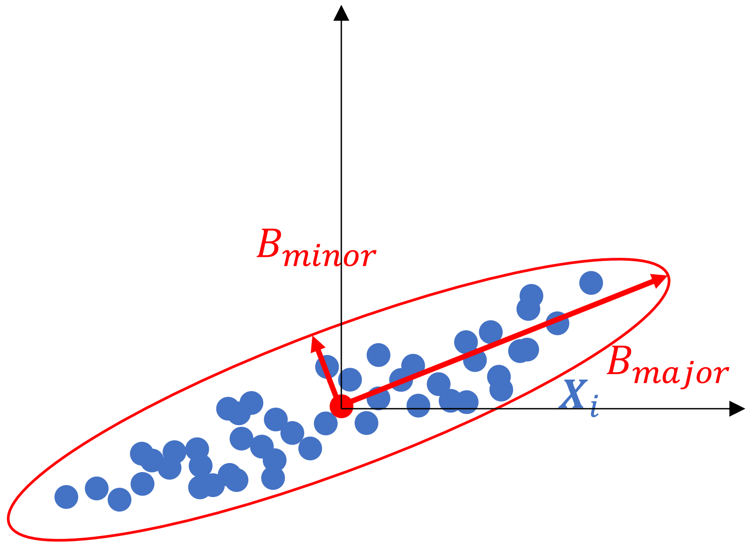
- Notice, variance is a special case of covariance

$$\sigma^2(X) = \mathrm{cov}(X,X)$$

- In 2D, we have 4 possible covariances, represented in the **covariance matrix**

$$\Sigma(X,Y) = \begin{bmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) \\ \mathrm{cov}(Y,X) & \mathrm{cov}(Y,Y) \end{bmatrix}$$





http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/

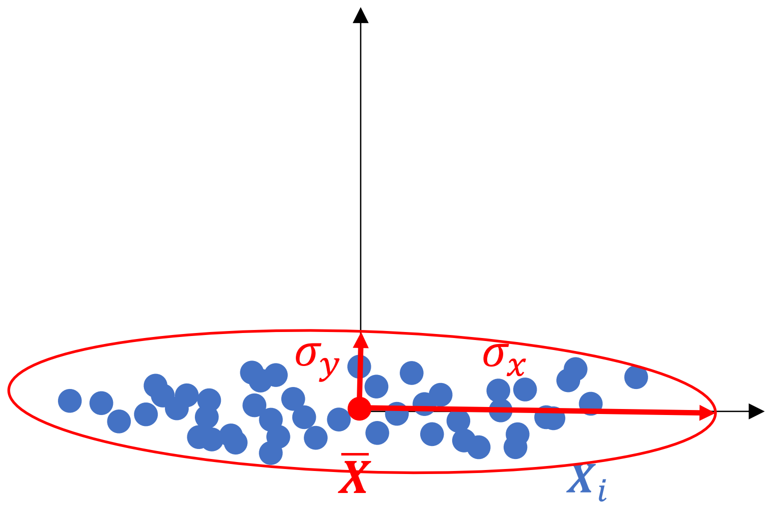
Covariance Matrix

- This is the **covariance matrix**

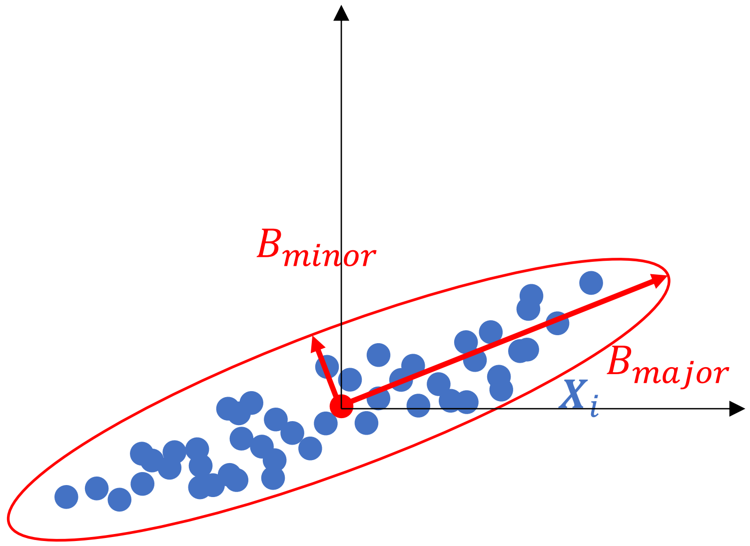
$$\Sigma(X,Y) = \begin{bmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) \\ \mathrm{cov}(Y,X) & \mathrm{cov}(Y,Y) \end{bmatrix}$$

- Note: this matrix is symmetric since $\mathrm{cov}(X,Y)= \mathrm{cov}(Y,X)$
- If our data is **uncorrelated**, the covariance matrix will be of the form:

$$\Sigma(X,Y) = \begin{bmatrix} \sigma_X^2 & 0 \\ 0 & \sigma_Y^2 \end{bmatrix}$$



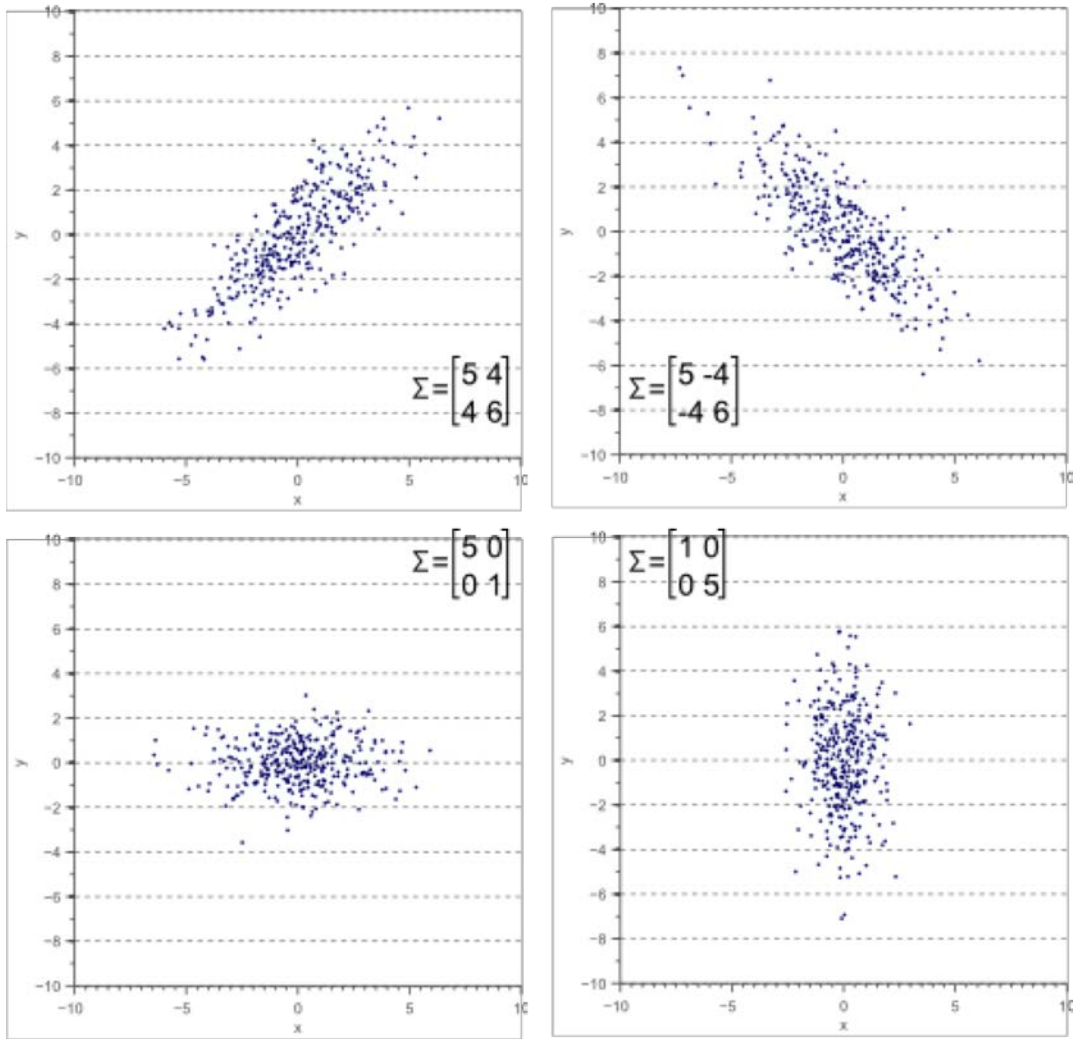
uncorrelated



correlated

Covariance Matrix - Intuition

- The off-diagonal terms of the covariance matrix give us an idea of the relationship of the data across dimensions
- Note that if the off-diagonal terms are zero, there is no obvious inter-dimension relationship!



http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/

Ellipsoid Equation

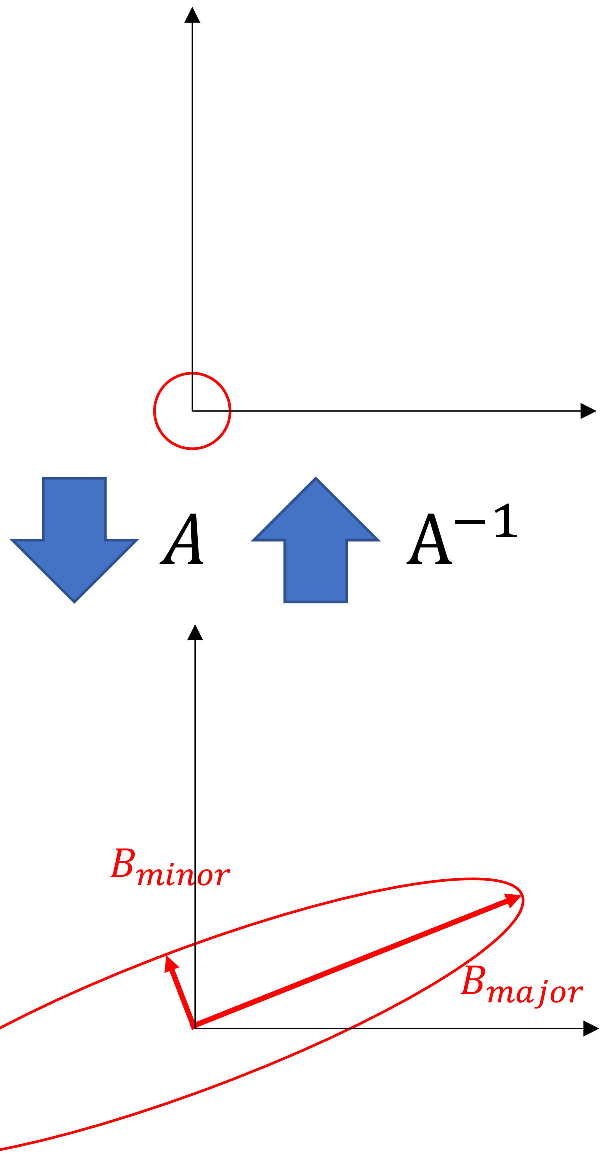
Ellipsoid (N-dim Ellipse) Equation:

$$(\boldsymbol{X}-\boldsymbol{\mu})^T \mathrm{A}^{-1}(\boldsymbol{X}-\boldsymbol{\mu}) = 1$$

where A^{-1} is an inverse transformation matrix ($N \times N$), and $\boldsymbol{X}$, $\boldsymbol{\mu}$ are N dimensional (col) vectors

- Here, the Ellipsoid is defined explicitly as a linearly transformed (scaled/rotated) unit circle/sphere:

$$(\boldsymbol{X}-\boldsymbol{\mu})^T\,(\boldsymbol{X}-\boldsymbol{\mu}) = r^2$$



https://datascienceplus.com/understanding-the-covariance-matrix/

https://www.xarg.org/2018/04/how-to-plot-a-covariance-error-ellipse/

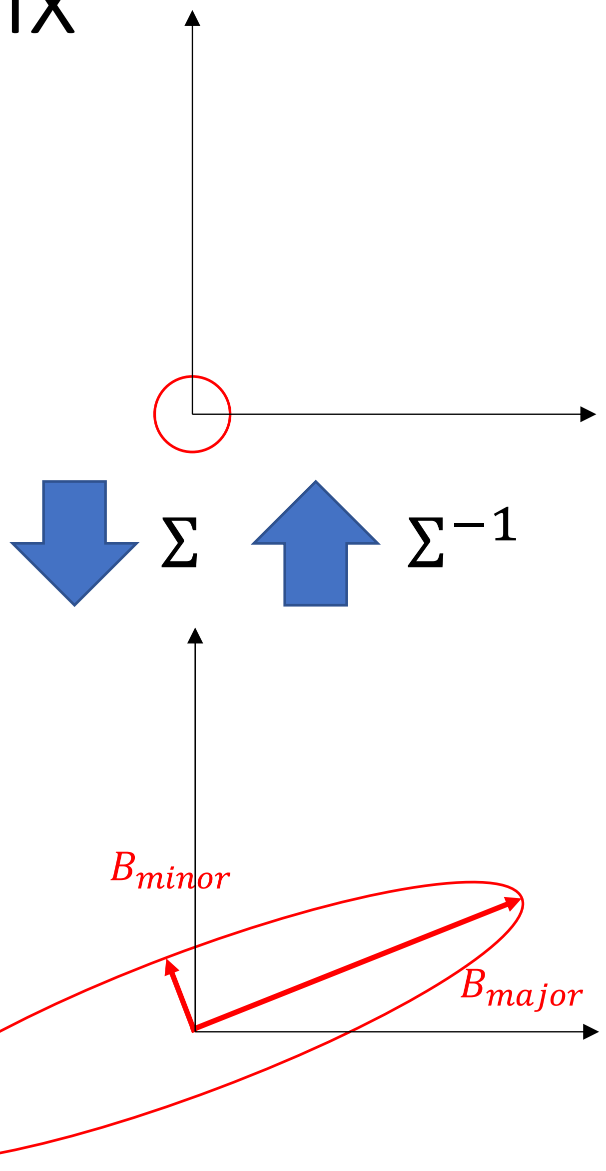
Covariance as a Transformation Matrix

Ellipsoid (N-dim Ellipse) Equation:

$$(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu}) = 1$$

where Σ^{-1} is the inverse covariance matrix ($N\times N$), and $\boldsymbol{X}$, $\boldsymbol{\mu}$ are N dimensional (col) vectors

- The **covariance $\boldsymbol{\Sigma}$** is actually a **linear transformation** telling us how our data differs from a dataset with no correlations
- What we are actually interested in however are the principle components/ellipse axes – how do we get those from our covariance matrix?



https://datascienceplus.com/understanding-the-covariance-matrix/

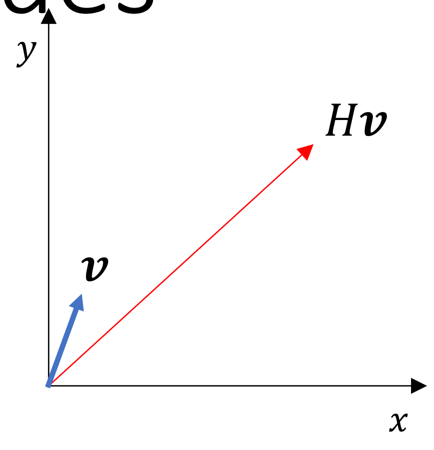
https://www.xarg.org/2018/04/how-to-plot-a-covariance-error-ellipse/

Math Review: Eigenvectors/Eigenvalues

- Definition: $\boldsymbol{v} \neq \mathbf{0}$ is an **eigenvector** of a matrix H if

$$H\boldsymbol{v} = \lambda\boldsymbol{v}$$

where λ is a scalar, called the **eigenvalue** of $\boldsymbol{v}$



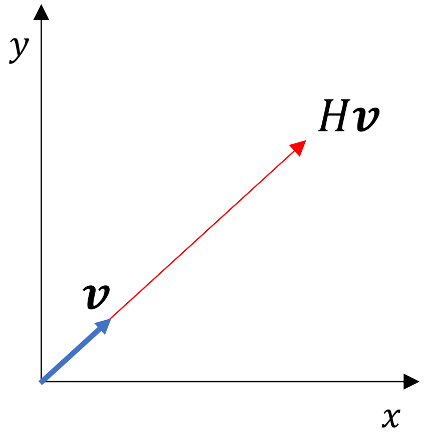
$\boldsymbol{v}$ is **not** an eigenvector

Geometric Intuition:

- A general transformation may be defined

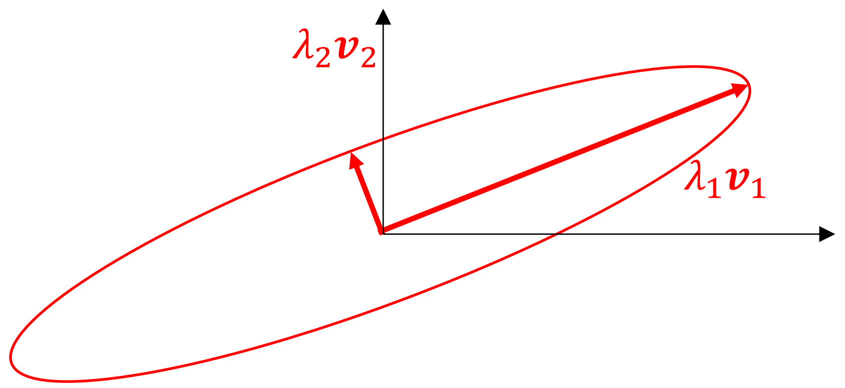
$$I = H\boldsymbol{v}$$

- $\boldsymbol{v}$ is an eigenvector of H if multiplication (transformation) by H preserves $\boldsymbol{v}$'s direction
- Vectors in the direction of the axis of rotation are unchanged in a transformation…



$\boldsymbol{v}$ **is** an eigenvector

Eigenvectors/values and Ellipse



- It turns out the eigenvectors/eigenvalues of our correlation matrix give us the direction/size of our ellipse axes!
  - Eigenvectors of A give us the basis (directions) of the ellipse's major/minor axis
  - Eigenvalues give us the size of the ellipse's major/minor axis
- Assume that we have eigenvectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_M$ such that $\lambda_1 > \lambda_2 > \cdots > \lambda_M$
  - $\boldsymbol{v}_1$ is the vector pointing in the direction of **largest variance**
  - $\boldsymbol{v}_M$ is the vector pointing in the direction of the **least variance**

Calculating Eigenvectors for Symmetric Matrix

- If Σ is a $M \times M$ symmetric matrix (like the covariance), then the singular value decomposition (SVD) of A:

$$\Sigma = U \Lambda U^T$$

where U is an $M \times M$ matrix of the eigenvectors as columns, and Λ is an $M \times M$ diagonal matrix with the eigenvalues as the diagonal, i.e.

$$\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_M), U = \begin{bmatrix} | & | & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 \cdots & \boldsymbol{v}_M \\ | & | & | \end{bmatrix}$$

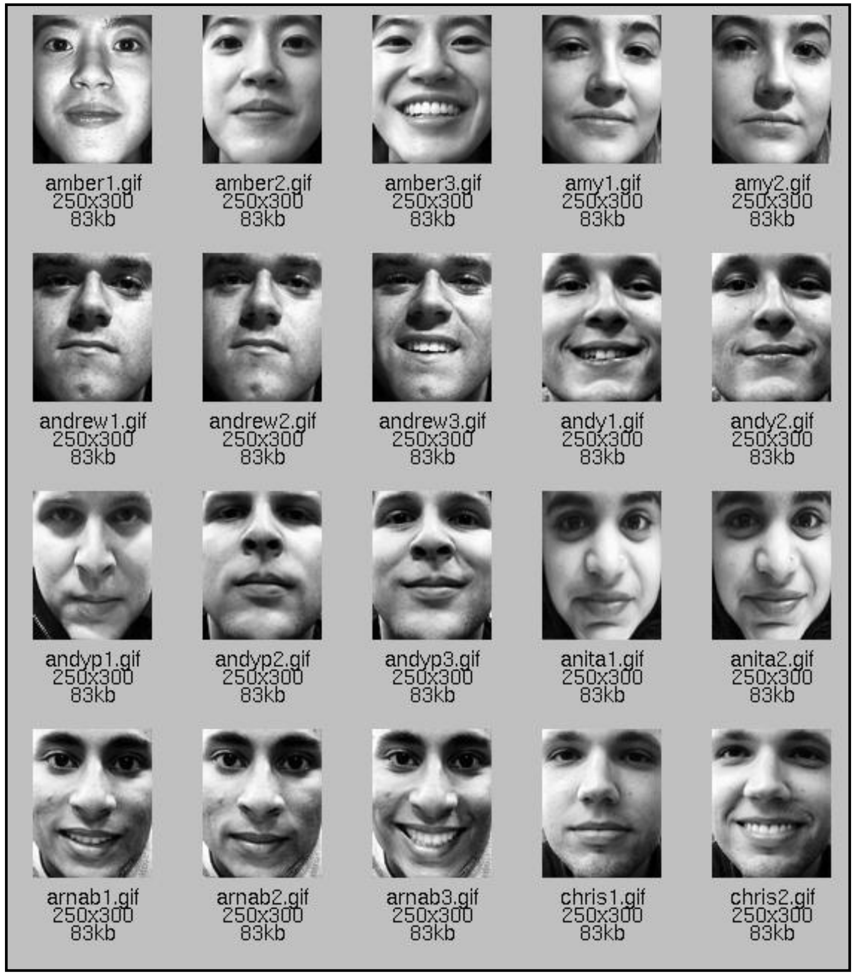
- There are many efficient implementations of SVD, so this is great!

Topic 6: 2D Images and Curves

- Images as Vectors
- Template Matching
- Cross-Correlation
- Template Matching using Cross-Correlation
- Dimensionality Reduction
- Principle Component Analysis (PCA)
- Understanding PCA
- **Case Study: EigenFaces**

PCA Application: EigenFaces

- EigenFaces uses PCA to recognize faces!
- Dataset: image patches of faces, dimensions 250×350 (75000-dimensional vectors)
- $\overline{\boldsymbol{X}}$ = "Mean" Face image
- $\boldsymbol{B}_1, \dots, \boldsymbol{B}_d$ (where $d < 20$): the "eigenfaces"
- Each face patch in the dataset can be represented as a linear combination of the "eigenfaces"



EigenFaces: Database Creation Algorithm

Given N face patches $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ of dimension M=75000:

1) Calculate mean of image patch vectors

$$\overline{\boldsymbol{X}} = \frac{1}{N}\sum \boldsymbol{X}_i$$

2) Subtract the mean from all patches (centre)

$$\boldsymbol{Z}_i = \boldsymbol{X}_i - \overline{\boldsymbol{X}}$$

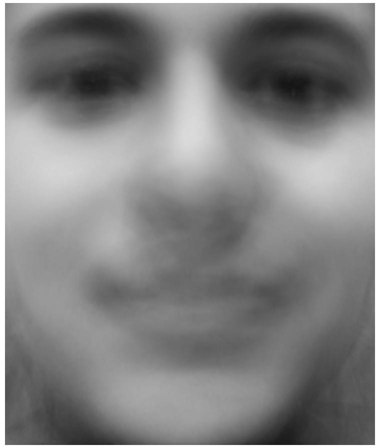
3) Create an $M \times N$ matrix of all centred patch vectors (arranged as columns of matrix)

$$Z = [\boldsymbol{Z}_1 \quad \boldsymbol{Z}_2 \cdots \quad \boldsymbol{Z}_N]$$

4) Find ***eigenvectors*** $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_d$ corresponding to the d (where $d < 15$) largest ***eigenvalues*** $\lambda_1, \ldots, \lambda_d$ of the correlation matrix ZZ^T

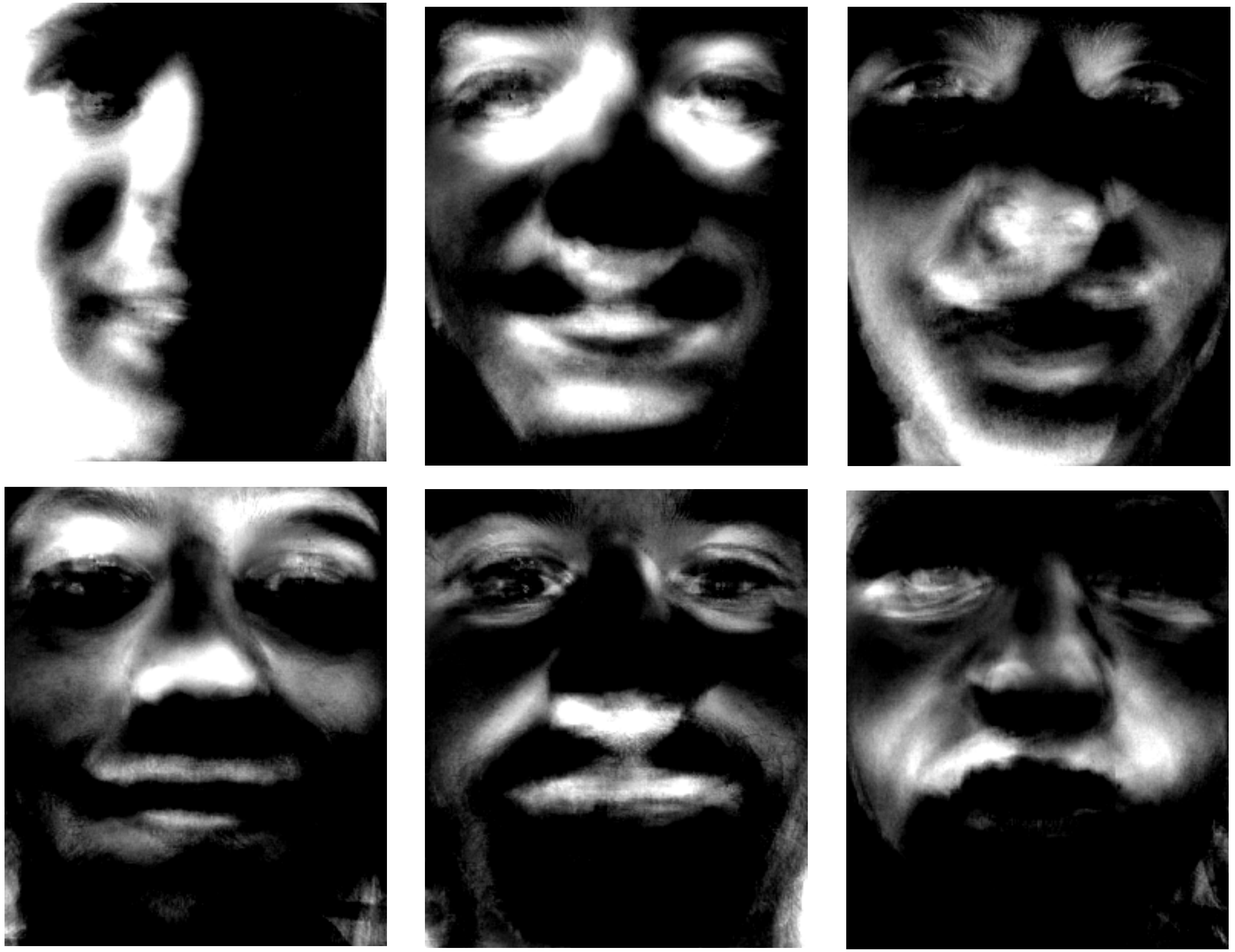
5) Store the eigenvectors $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_d$, mean image $\overline{\boldsymbol{X}}$ and vectors in new d-dimensional basis $\boldsymbol{Y}_i = [Y_i^1 \quad Y_i^2 \cdots \quad Y_i^d]$

EigenFaces



“Mean” Face

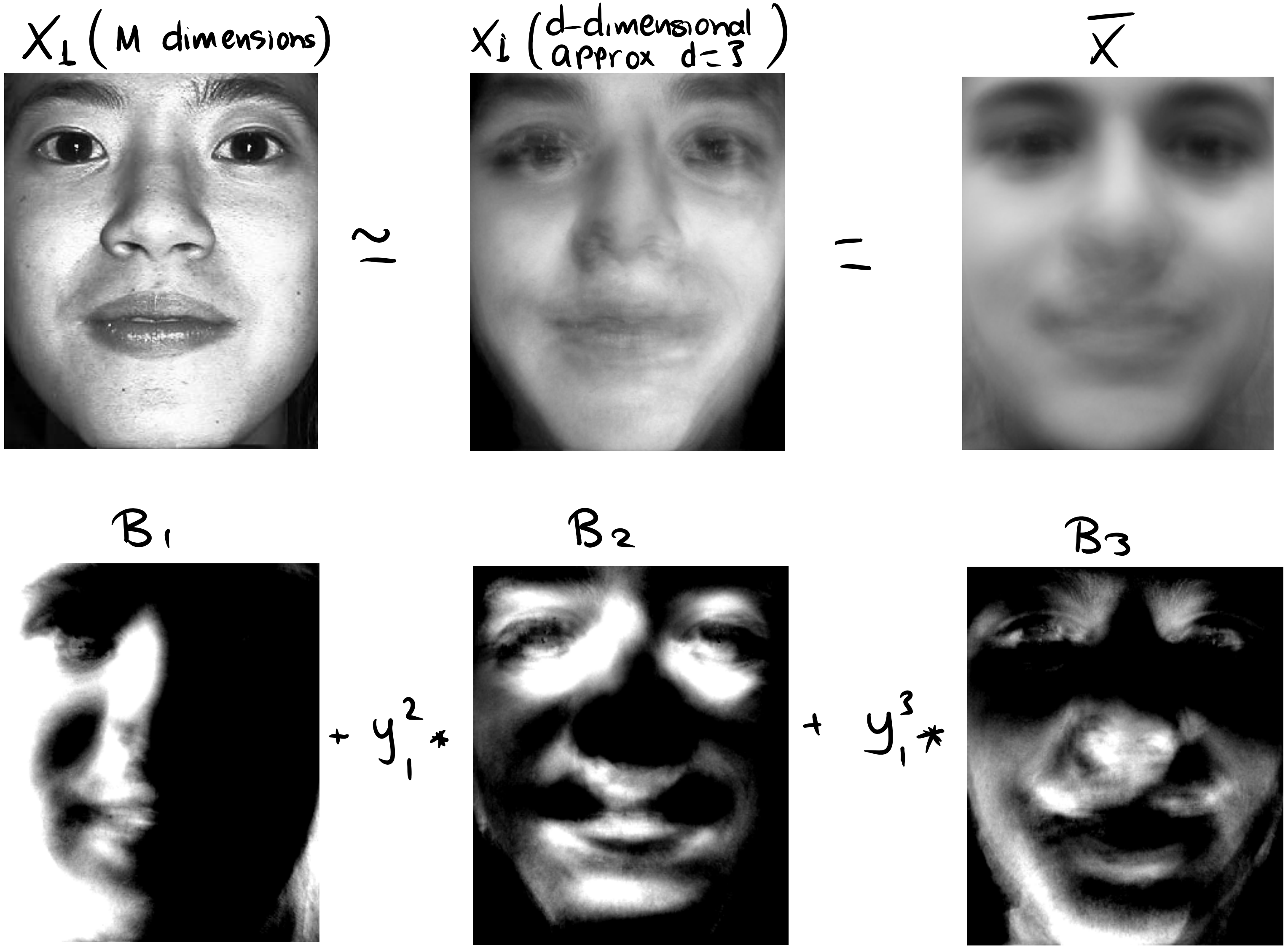
(not so mean!)



Top-6 Eigenvectors

EigenFaces: Representing a Face

- We can represent any face as a linear combination of the basis vectors.
- Not very flattering, but consider this image is represented as only 3 numbers in the database!
- Storage for N faces:
  Images: $75000N$
  EigenFaces: $4 \cdot 75000 + 3N$



EigenFaces: Recognition

Given a query image $\boldsymbol{T}$ and our EigenFaces database

1. Compute coordinates of $\boldsymbol{T}$ in the EigenFaces basis, i.e. j^th coord:

$$S_i^j = \boldsymbol{B}_j^T T$$

2. Find the vector $\boldsymbol{Y}_i$ in the database that is closest to $\boldsymbol{S}_i$
3. Return face image $\boldsymbol{X}_i$, i.e. the vector in the original image space

End of Topic 6