

Appearance Matching

Shree K. Nayar

Monograph: FPCV-5-3

Module: Perception

Series: First Principles of Computer Vision

Computer Science, Columbia University

May, 2025

[FPCV Channel](#)

[FPCV Website](#)

<h1>Appearance Matching</h1> <p>Shree K. Nayar Columbia University</p> <p>Topic: Appearance Matching, Module: Perception First Principles of Computer Vision</p> <p>1</p>	<h1>Appearance Matching</h1> <hr/> <p>3D object representation and recognition using visual appearance.</p> <p>Topics:</p> <ul style="list-style-type: none">(1) Shape vs. Appearance(2) Learning Appearance(3) Principal Component Analysis(4) Parametric Appearance Representation(5) Appearance Matching(6) Example Applications <p>2</p>
---	---

This lecture is about recognizing a three-dimensional object from its appearance in a two-dimensional image. This paradigm is known as appearance matching. We first discuss two different approaches to representing a 3D object. The first approach is the explicit representation of the 3D geometry (shape) of the object. We will discuss the pros and cons of this approach, and argue that a more effective approach to object recognition is to represent the visual appearance of an object, rather than its shape.

The appearance of an object in an image depends on several intrinsic parameters (the object's 3D shape and surface reflectance) and extrinsic parameters (the pose of the object and its illumination). Our approach to appearance representation is to take a large number of images of the object by varying its extrinsic parameters. The goal then is to efficiently present this large set of images for use during recognition.

Since each image in the object's image set has a large number of pixels, and we have a large number of images of the object, a key challenge with appearance matching is to compactly represent this high-dimensional data in a low-dimensional space. This reduction in dimensionality is achieved using principal component analysis (PCA), which is a technique that is widely used in many different disciplines of science and engineering. We describe PCA and its use for appearance representation. The end result is an appearance model of the object that is parameterized by the extrinsic parameters (pose and illumination) of the object.

Based on this parametric appearance representation, we develop an efficient algorithm for recognizing an object from its appearance in an image, and determining its extrinsic parameters. Finally, we will discuss a few applications of appearance matching, including, face recognition, general 3D object recognition, visual tracking and servoing in robotics, and visual inspection.

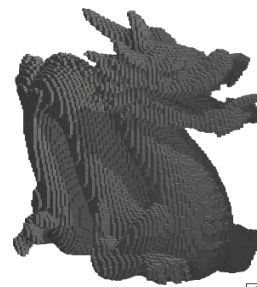
Shape vs. Appearance

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

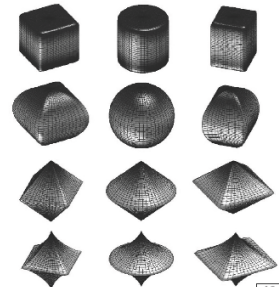
3

Shape Representations



Voxel Representation

1.1



Analytical Representation

1.2

(Ex: Superquadrics, $|x|^r + |y|^s + |z|^t = 1$)

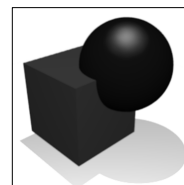
4

Let us discuss the two approaches to representing a 3D object—shape and appearance. Shape representation has a long history. The explicit representation of the 3D geometry of an object is used in many domains such as computer graphics, manufacturing, and factory automation. Shown on the left in slide 4 is the voxel representation of an object, where a voxel is a generalization of a pixel. A voxel is a volumetric element, i.e., a cell in 3D. The object is represented by simply identifying which cells are occupied. Voxel representations have been useful in computer graphics and computer-aided design.

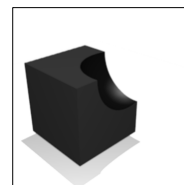
Another way to represent the 3D geometry of an object, in particular an opaque object, is by representing its surface. The surface can be represented using simple primitives like planes, spheres, and low-order polynomials or more complex primitives such as splines. Shown on the right is a representation called the superquadric. The equation of the superquadric is $|x|^r + |y|^s + |z|^t = 1$. The parameters r , s , and t are real numbers (not just integers) and by simply varying them we can generate all of the shapes shown on the right.

In the field of constructive solid geometry (CSG), an object is represented as the result of applying Boolean operations to simple primitive shapes such as spheres, cubes, and cones. Shown here is the union, difference, and intersection of a sphere and a cube. The field of CSG is the workhorse of application domains such as product design and manufacturing.

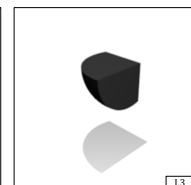
Shape Representations



Union



Difference



Intersection

1.3

Constructive Solid Geometry (CSG)

5

The use of shape representations for object recognition poses a few challenges. Firstly, it requires the explicit representation of the 3D shape of each object, which entails either designing a 3D model by hand or measuring an object using a 3D sensor (such as one based on active illumination). This is a tedious process when dealing with large numbers of objects. In addition, the recognition procedure may require the use of a 3D sensor to recover the structure of the scene that includes the objects of interest.

Issues with 3D Shape Matching

Requires measuring of 3D shapes for:

- Creating the database of object shapes (Offline)
- Recognition (Online)

Can be computationally expensive for large databases.

6

An attractive approach to object recognition would be to directly recognize objects from their appearance in an image. We can describe the visual appearance of an object as a function of its intrinsic and extrinsic parameters. The intrinsic parameters include the object's 3D shape and reflectance properties (BRDF). In the case of a rigid object, its intrinsic parameters do not change with time. The extrinsic parameters of an object include its 3D pose and its illumination. For recognition, we aim to compactly represent an object using the above definition of visual appearance.

Visual Appearance



$$\text{Visual Appearance} = \mathcal{F} \left\{ \begin{array}{l} \text{Shape} \\ \text{Reflectance} \end{array} \right\} \quad \text{Intrinsic Parameters}$$

$$\left\{ \begin{array}{l} \text{Pose} \\ \text{Illumination} \end{array} \right\} \quad \text{Extrinsic Variables}$$

7

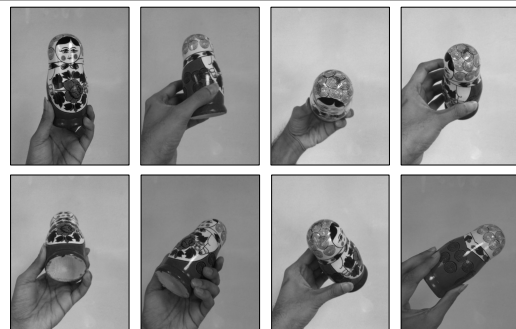
Learning Appearance

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

8

Appearance Learning by Humans

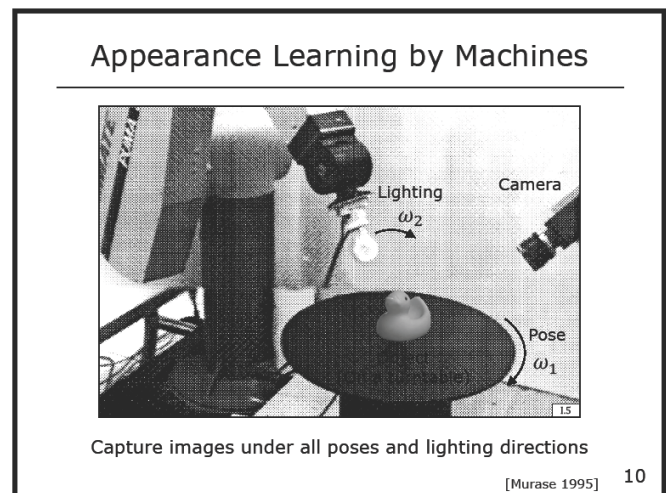


We learn the appearance of an object by looking at it from a different directions.

9

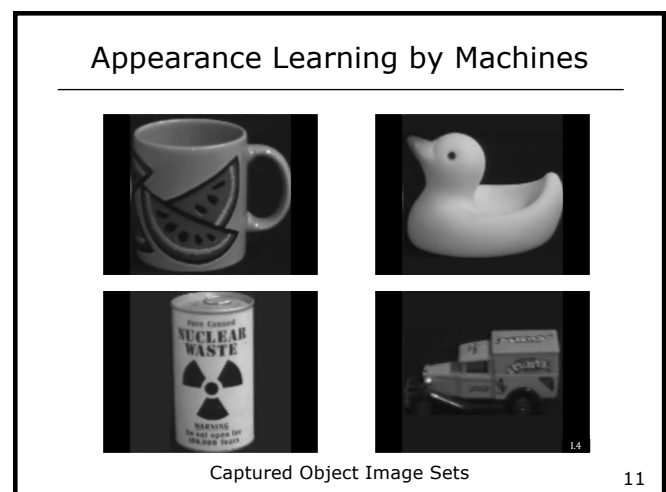
Consider a situation wherein you are given the doll on the right and asked to memorize it. You will most likely look at the doll from different perspectives. These perspectives are used to create a mental model of the doll, albeit one that we don't fully understand. When the doll is shown to you in the future, you are able to recognize it. We will take a similar approach in the context of appearance matching for computer vision.

When you drop an object on the ground, it lands in one of a few configurations, each one called a stable configuration. The rubber duck seen here is sitting on a turntable in one of its stable configurations. Let the angle ω_1 represent the pose (rotation) of the object in this stable configuration. The pose of the duck can be varied by spinning the turntable. Assume that in our recognition scenario, the illumination could vary. To account for this, we can vary the angle ω_2 of the light source by using the robot it is mounted on. Therefore, a large number of images of the duck, corresponding to different poses and illuminations, can be captured using the turntable and the robot. This is the learning image set corresponding to the object. Note that while we are using a single parameter (ω_1) to represent pose and a single parameter for illumination (ω_2), in general, pose and illumination could each involve multiple parameters.



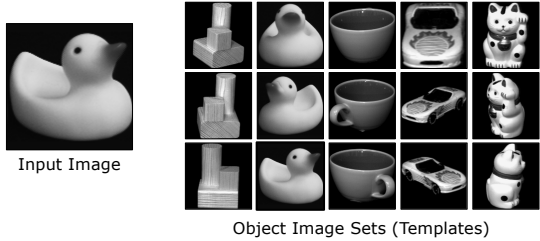
From each image in an object's learning image set, the object is segmented and the segmented region is scaled such that the larger of its two dimensions equals a canonical size. Shown here are segmented and scaled images of four different objects.

We will assume that in each image used for learning and recognition, the object has a uniform background (black in the examples shown here). This is because we will be using the entire image for both learning and recognition and hence require the image not to include variations due to the background. In addition, we will assume that the object can be segmented in full and is not occluded by other objects.



Given our object image sets, what would be the simplest way to recognize a novel input image, such as the duck image on the left? The most naïve way is to perform template matching. Given a segmented and scaled input image, we could use it as a template and match it with each and every image in our learning set. As discussed in previous lectures, the matching can be done using the sum of absolute differences (SAD), the sum of squared differences (SSD), or normalized cross-correlation (NCC) which can handle differences in brightness and camera gain. In a typical recognition system, template matching may have to be done with thousands of templates for thousands of objects. Clearly, this is computationally prohibitive for virtually any real-world application.

Naïve Recognition: Template Matching



Input Image

Object Image Sets (Templates)

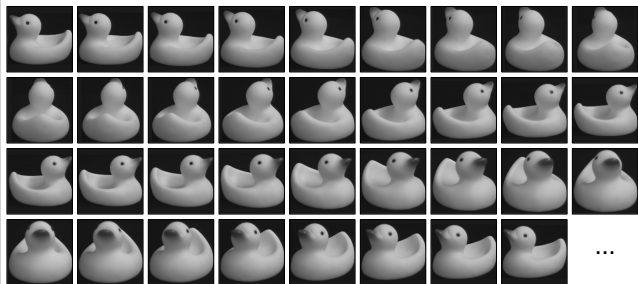
Perform Template Matching using SAD/SSD/NCC as similarity measure.

Template Matching with thousands of "templates" for potentially millions of objects is very expensive and time consuming.

12

Our goal is to perform the above matching, but in a very efficient manner. Shown here is the learning image set of the duck. Note that there is great similarity between any two consecutive images in the set. That is, the image set is highly redundant. We seek to exploit this redundancy to achieve a compact representation of the object's appearance.

Object Image Set



Exploit redundancy to achieve a compact appearance representation that makes matching efficient.

13

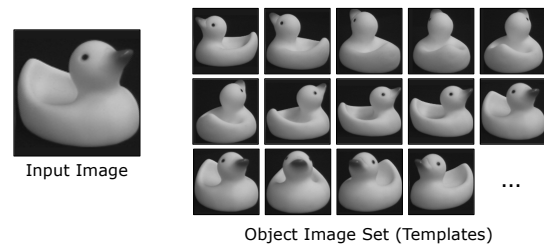
Principal Component Analysis

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

14

Representing Appearance

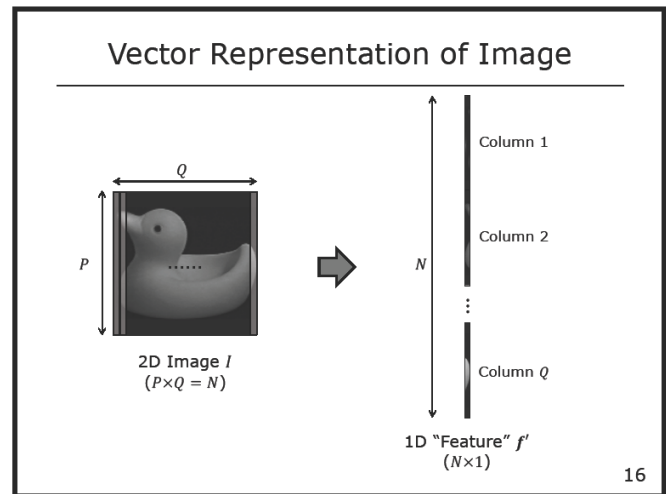


We want to transform the images into a different "space" or "domain", where matching one image with another is more efficient.

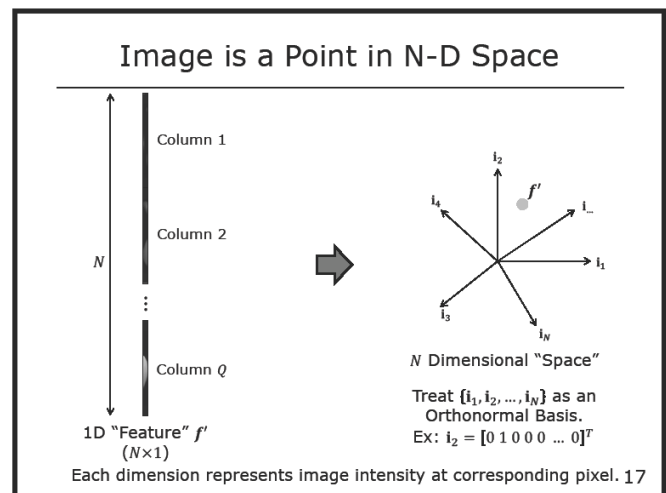
15

The learning image set of an object can be viewed as features in a high-dimensional space, where the dimensionality of the space equals the number of pixels in the image. Our goal is to exploit the redundancy in the image set discussed previously to represent the image set in a lower-dimensional space. This reduction in dimensionality is achieved using principal component analysis (PCA). The end result is a representation that allows us to efficiently perform template matching between images.

Here is an image with P rows and Q columns, i.e., a total of N pixels. The image can be represented as an $N \times 1$ vector \mathbf{f}' by stacking the columns of the image, as shown on the right. This mapping of an image to a vector can be done in many ways. We could have stacked the rows of the images instead of the columns, or even randomly mapped image pixels to vector elements. The only requirement with mapping the image to a vector is that the same mapping needs to be used for all images during learning as well as recognition.

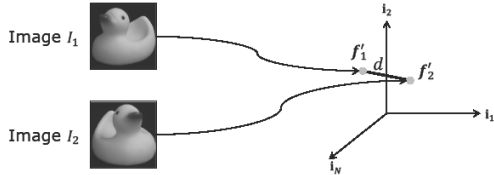


Shown on the left is the $N \times 1$ feature vector \mathbf{f}' . We can construct a N -dimensional space, shown on the right, where each of the dimensions corresponds to the brightness of the image at the corresponding pixel. For instance, the unit vector \mathbf{i}_1 corresponding to the first dimension is used to represent the brightness of the first pixel, \mathbf{i}_2 corresponds to the second pixel, and so on. Thus, in this N -dimensional space, our image feature vector \mathbf{f}' is a point. This N -dimensional space is referred to as an orthonormal basis because the unit vectors \mathbf{i}_n are orthogonal to each other. For instance, \mathbf{i}_2 is a unit vector with a 1 in its second element because it corresponds to the second pixel, and 0 elsewhere.



The representation of an image as a point in N -dimensional space gives us a different perspective on template matching. From our lectures on image processing, template matching between two images, I_1 and I_2 , is done by finding their correlation, which is the sum of squared difference (SSD) between the images [1]. When the two images are represented as points f'_1 and f'_2 in N -dimensional space, correlation between the images is equal to the L^2 distance between the two points [2].

Interpretation of Template Matching



Correlation in Image Space Square of L^2 Distance in N-D

$$SSD = \sum_p \sum_q (I_1[p, q] - I_2[p, q])^2 \quad \equiv \quad d^2 = \|f'_1 - f'_2\|^2$$

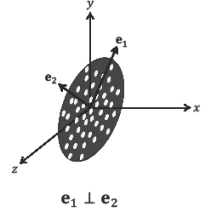
[1] [2]

18

Now, let us discuss the problem of dimensionality reduction. Consider this distribution of points in a three-dimensional space. The points lie on a two-dimensional plane within the three-dimensional space. This suggests that it is redundant to represent each of these points using the three coordinates x , y , and z . Since they lie on a plane, we can define a new coordinate system using two orthogonal vectors, e_1 and e_2 , that lie on the two-dimensional plane in which all the points lie. Then, we only need two coordinates (distances along e_1 and e_2) to represent each point. This is a simple example of dimensionality reduction where we have gone from three dimensions to two dimensions.

Dimensionality Reduction

Consider a distribution of points in 3D space that actually lie on a 2D plane.



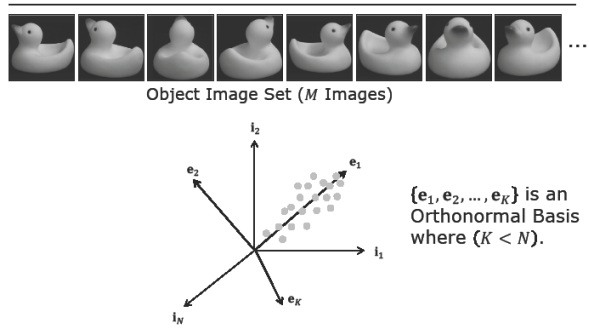
It is redundant to represent each point with 3 coordinates.

If we use a new 2D coordinate system $\{e_1, e_2\}$ that lies on the plane of distribution, each point can be represented with just 2 coordinates.

19

Now, consider an object image set with M images consisting of N pixels each. In N -dimensional space, we have a distribution of points as shown here, which we will refer to as an appearance distribution. Due to the similarity between the images in the object image set, the distribution of points in the N -dimensional space can be assumed to be highly structured. As a result, these points reside in a K -dimensional space, where $K < N$. This is the key observation that we want to exploit.

Dimensionality Reduction



Object Image Set (M Images)

$\{e_1, e_2, \dots, e_K\}$ is an Orthonormal Basis where $(K < N)$.

It is possible to express the feature point distribution in a Lower Dimensional Space ($K < N$) with good accuracy.

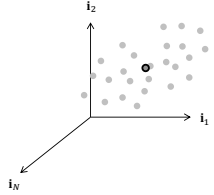
20

This K -dimensional space can be represented by K unit vectors $\{e_1, e_2, \dots, e_K\}$ that are orthogonal to each other and form an orthonormal basis. Our hope is that K is much smaller than N . Our goal then is to find the orthonormal basis $\{e_1, e_2, \dots, e_K\}$. This is achieved using principal component analysis.

The first step in reducing dimensionality is to compute the mean \mathbf{c} of the M images $\{f'_1, f'_2, \dots, f'_M\}$ in our object image set [1]. Then, we shift the origin of the coordinate frame by subtracting the mean image \mathbf{c} from each one of the original images [2]. By moving the origin of the coordinate frame to the mean, we are in a better position to measure the variability of points within the distribution.

Subtracting the Mean

Given M images $\{f'_1, f'_2, \dots, f'_M\}$ of an object, the Mean Image is:

$$\mathbf{c} = \frac{1}{M} \sum_{m=1}^M f'_m \quad [1]$$


Subtract the mean from the object image set so as to move the origin of the new basis to the centroid of the distribution:

$$f_m = f'_m - \mathbf{c} \quad [2]$$

21

The first principal component, e_1 , represents the direction along which the points in the distribution have maximum variance. In other words, if we take all the points in the distribution and project them onto the first principal component e_1 , the variance of the projections is going to be maximum, i.e., there is no other vector in the N -dimensional space for which the variance is greater.

It turns out that if we use the least squares method to find a line that best fits our distribution, its direction is the same as the direction of e_1 . We refer to e_1 as the first principal component. If we wanted to represent each point f (an image) in a one-dimensional space, its coordinate p in the space can be found as the dot product of f with e_1 [1].

Principal Components

1st Principal Component (e_1):

The direction of maximum variance in the image set.
(Equivalent to Least Squares Fitting of a Line)

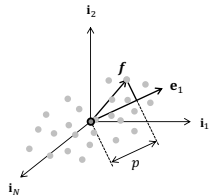


Image Representation:
Project the image f onto the principal component e_1 .

$$p = e_1 \cdot f \quad [1] \quad (\text{Dot product})$$

Image is represented by a single number: p

22

Clearly, representing each image with a single number is not going to be adequate for most applications. Hence, we will look at the second principal component, e_2 . Since we are looking for an orthonormal basis, e_2 must be orthogonal to the first principal component e_1 . We know that there are an infinite set of vectors that are orthogonal to e_1 . However, e_2 is the vector in this infinite set along which the variance of the distribution is maximum. Now we have a more descriptive representation of each image. An image f can now be represented by a two-dimensional point p whose coordinates (p_1, p_2) can be found by projecting f onto the principal components e_1 and e_2 , respectively [1].

The above gives us intuition for what principal components mean. We will soon describe how these principal components are computed. Let us assume for now that we have found K principal components $\{e_1, e_2, \dots, e_K\}$. Then, each image f can be represented by a point p in K -dimensional space [1]. Our hope is that each image is well represented within this K -dimensional space, and because of the structure of our distribution, K is much smaller than N . When this is the case, we have a significant reduction in dimensionality.

The principal components $\{e_1, e_2, \dots, e_K\}$ are referred to as a linear subspace. Given such a subspace, we can use it for two purposes. First, we can project an image f to the subspace to obtain the lower-dimensional representation p using the forward projection equation [1]. Second, we can use p to reconstruct the original image using the backward projection equation [2]. Note that this reconstruction is only an approximation of the original image f because we are bound to lose information while going from N to K dimensions.

Principal Components

2nd Principal Component (e_2):

The direction of 2nd maximum variance in the image set such that $e_1 \perp e_2$.

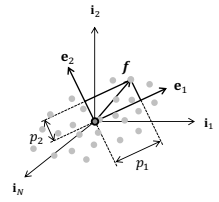


Image Representation:

Project the Image f onto the principal components $\{e_1, e_2\}$.

$$p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = [e_1 \ e_2]^T f \quad [1]$$

Image is represented in 2 dimensions.

23

can be found by projecting f onto the principal

Principal Components

K^{th} Principal Component (e_K):

The direction of K^{th} maximum variance in the image set such that $e_1 \perp e_2 \perp \dots \perp e_K$.

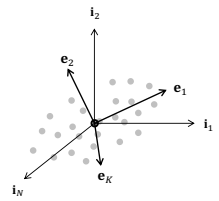


Image Representation:

Project the Image f onto the principal components $\{e_1, e_2, \dots, e_K\}$.

$$p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_K \end{bmatrix} = [e_1 \ e_2 \ \dots \ e_K]^T f \quad [1]$$

Image is represented in $K < N$ dimensions.

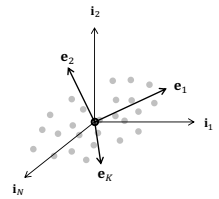
24

Forward and Back Projection

Forward Projection:

[1]

$$p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_K \end{bmatrix} = [e_1 \ e_2 \ \dots \ e_K]^T f$$



Back Projection:

$$f \approx p_1 e_1 + p_2 e_2 + \dots + p_K e_K$$

[2]

$$f \approx \sum_{k=1}^K p_k e_k$$

$\{e_1, e_2, \dots, e_K\}$ is referred to as Linear Subspace.

25

Finding Principal Components

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

26

Finding the Principal Components

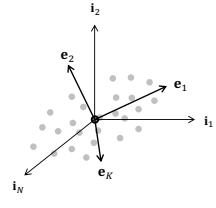
Given: Mean-Subtracted Image Set
 $\{f_1, f_2, \dots, f_M\}$ (f_m is $N \times 1$ vector)

Find: Orthonormal Basis $\{e_1, e_2, \dots, e_K\}$
(e_k is $N \times 1$ vector)

Such that: $f_m \approx \sum_{k=1}^K p_k^{(m)} e_k$ [1]

Where: $p_k^{(m)} = e_k^T f_m$ (Linear) [2]

$e_i^T e_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$ (Orthonormal) [3]



27

Let us now look at how we can find the principal components. Given an image set, we will convert it to a mean-subtracted image set $\{f_1, f_2, \dots, f_M\}$. Our aim is to find the orthonormal basis $\{e_1, e_2, \dots, e_K\}$ which corresponds to the K principal components. This orthonormal basis allows us to approximate an original image f_m as a linear combination of the principal components e_k , where the weights of the combination are the projections $p_k^{(m)}$ [1]. The projections $p_k^{(m)}$ are obtained from the forward projection, which corresponds to finding the dot product of image f_m with each of the principal components [2]. Since the principal components form an orthonormal basis, they are unit vectors that are orthogonal to each other [3].

Let us formulate the problem of finding the principal components. We know that the first principal component corresponds to the direction along which the distribution of our images has maximum variance. Let us treat the images in our set as a random variable f , and let vector e represent the first principal component. We want to find the e that maximizes the variance of the dot product of f and e . We can rewrite this variance as the expected value of the square of $f \cdot e$ minus the expected value of $f \cdot e$ [1]. Since e is a constant and not a random variable, we get [2].

The expected value of an image $E[f]$ is zero because the images are mean-subtracted. This simplifies the equation to the expected value of the square of $e \cdot f$. Since both e and f are vectors, we can represent $\{e \cdot f\}^2$ as $e^T f f^T e$. Once again, since e is not a random variable, we can pull it out of the

Finding the Principal Components

Projection of the Random Variable (Image) f along the 1st Principal Component e is $(f \cdot e)$.

Find e that Maximizes Variance of $(f \cdot e)$.

By Definition: $\text{Var}[f \cdot e] = E[\{(f \cdot e) - E[f \cdot e]\}^2]$ [1]

where $E[X]$ is the Expected Value of a random variable X .

$$\begin{aligned} \text{Var}[f \cdot e] &= E[\{e \cdot (f - E[f])\}^2] \quad [2] \quad (\text{Mean of } f: E[f] = 0) \\ &= E[\{e \cdot f\}^2] = E[e^T f f^T e] = e^T E[f f^T] e \end{aligned}$$

$\text{Var}[f \cdot e] = e^T R e$ [3] where $R_{N \times N} = E[f f^T]$ is Covariance Matrix

MATH PRIMER

28

expectation. Given that we are dealing with discrete images, $E[\mathbf{f}\mathbf{f}^T]$ is an $N \times N$ matrix R , called the covariance matrix. Thus, the variance of $\mathbf{f} \cdot \mathbf{e}$ is $\mathbf{e}^T R \mathbf{e}$ [3].

We want to find the \mathbf{e} that maximizes $\mathbf{e}^T R \mathbf{e}$ such that \mathbf{e} is a unit vector, i.e., $\mathbf{e}^T \mathbf{e} = 1$. To this end, we define an objective function L , which is equal to $\mathbf{e}^T R \mathbf{e}$ minus a multiplier λ times $(\mathbf{e}^T \mathbf{e}) - 1$. To find the \mathbf{e} that maximizes L , we can find the derivative of L with respect to \mathbf{e} and set it to zero. The result is [1], which implies that the first principal component \mathbf{e}_1 is the eigenvector of R corresponding to the largest eigenvalue. In fact, the entire set of principal components $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_K\}$ that constitute the orthonormal basis we are looking for are the K eigenvectors of R corresponding to the largest K eigenvalues.

Finding the Principal Components

Projection of the Random Variable (Image) \mathbf{f} along the 1st Principal Component \mathbf{e} is $(\mathbf{f} \cdot \mathbf{e})$.

Find \mathbf{e} that Maximizes Variance of $(\mathbf{f} \cdot \mathbf{e})$:

$$\max_{\mathbf{e}} (\mathbf{e}^T R \mathbf{e}) \text{ such that } \mathbf{e}^T \mathbf{e} = 1$$

Maximize Objective function $L(\mathbf{e}, \lambda)$:

$$L(\mathbf{e}, \lambda) = \mathbf{e}^T R \mathbf{e} - \lambda(\mathbf{e}^T \mathbf{e} - 1)$$

Taking derivatives of $L(\mathbf{e}, \lambda)$ w.r.t \mathbf{e} and equating to 0:

$$[1] \quad R\mathbf{e} - \lambda\mathbf{e} = \mathbf{0} \quad [R\mathbf{e} = \lambda\mathbf{e}] \quad \text{Eigenvalue Problem}$$

First Principal Component is eigenvector corresponding to maximum eigenvalue.

29

Let us look at how the above principal component analysis (PCA) is applied in practice. First, we construct a $N \times M$ data matrix F which consists of images \mathbf{f}_1 through \mathbf{f}_m stacked together as columns [1]. The $N \times N$ covariance matrix R is computed as FF^T [2]. We then find the eigenvalues and eigenvectors of R . The first K eigenvectors correspond to the orthonormal basis in which we will represent all our images. This basis is also referred to as a linear subspace or an eigenspace.

There is an important question we have yet to answer. What dimensionality K is adequate for representing an image set? We will revisit this question shortly.

Principal Component Analysis (PCA)

$$\text{Data Matrix: } F = [\mathbf{f}_1 \quad \mathbf{f}_2 \quad \dots \quad \mathbf{f}_M] \quad (N \times M) \quad [1]$$

$$\text{Covariance Matrix: } R = FF^T \quad (N \times N) \quad [2]$$

Solve Eigenvalue Problem: $R\mathbf{e} = \lambda\mathbf{e}$

Eigenvalues: $\{\lambda_1, \lambda_2, \dots, \lambda_K\}$

Eigenvectors: $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_K\}$

Orthonormal Basis
Linear Subspace
"Eigenspace"

30

PCA and SVD

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

31

Singular Value Decomposition (SVD)

For any matrix A there exists a factorization:

$$A_{N \times M} = U_{N \times N} \cdot \Sigma_{N \times M} \cdot V^T_{M \times M}$$

where U and V^T are orthonormal and Σ is diagonal.

$$\Sigma_{N \times M} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & 0 & \dots & 0 \\ 0 & 0 & 0 & \sigma_4 & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & \sigma_M \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \end{bmatrix} \quad \sigma_1, \dots, \sigma_M: \text{Singular Values}$$

MATH PRIMER

32

We have discussed how we can use PCA to achieve dimensionality reduction. It turns out that there is a close relationship between PCA and a technique we discussed earlier in the context of structure from motion called singular value decomposition (SVD), which is widely used in many different domains. SVD states that any $N \times M$ matrix A can be factorized into the product of an $N \times N$ matrix U , an $N \times M$ matrix Σ , and a $M \times M$ matrix V^T . The matrices U and V are orthonormal matrices, and matrix Σ is a diagonal matrix. The diagonal elements of Σ are called singular values. These diagonal elements are non-negative making Σ a positive semidefinite matrix. The singular values are ordered in decreasing order of magnitude along the diagonal, with σ_1 being the largest value. As discussed earlier in the lecture on structure from motion, the number of non-zero singular values correspond to the rank of the matrix.

Let us apply SVD to our $N \times M$ data matrix F to factorize it into U , Σ , and V^T . We can then construct the covariance matrix R which is equal to FF^T by substituting $U \cdot \Sigma \cdot V^T$ for F [1].

Since V is an orthonormal matrix, V^T is equal to the inverse of V . Thus, $V^T V$ is the identity matrix I . We can therefore rewrite [1] as the product of U , a new matrix Λ , and U^T . Λ is a diagonal matrix since it is the product of two diagonal matrices, Σ and Σ^T .

SVD and Covariance Matrix R

Factorization of Data Matrix F :

$$F_{N \times M} = U_{N \times N} \cdot \Sigma_{N \times M} \cdot V^T_{M \times M}$$

Covariance Matrix: $R = FF^T$

$$R = U \cdot \Sigma \cdot \underbrace{V^T \cdot V}_{I} \cdot \Sigma^T \cdot U^T \quad [1]$$

$$R = U \cdot \Lambda \cdot U^T$$

33

Λ is an $N \times N$ diagonal matrix, where each diagonal element λ_i is equal to the square of the corresponding singular value σ_i of F . It turns out that the diagonal elements λ_i of Λ are the N eigenvalues of the correlation matrix R , ordered in descending order of magnitude. The columns of the matrix U are the N eigenvectors of R , i.e., the principal components, ordered in descending order of importance from left to right. By choosing the first K of these eigenvectors we have the K -dimensional subspace we are looking for. This is the relationship between PCA and SVD.

Eigenvalues and Eigenvectors of R

Factorization of Data Matrix F :

$$R_{N \times N} = U_{N \times N} \cdot \Lambda_{N \times N} \cdot U_{N \times N}^T$$

Eigenvalues:

$$\Lambda_{N \times N} = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & \lambda_K & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \lambda_{K+1} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \lambda_N \end{pmatrix} \quad \lambda_i = \sigma_i^2$$

Eigenvectors:

$$U_{N \times N} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_K \dots \mathbf{e}_N]^T$$

34

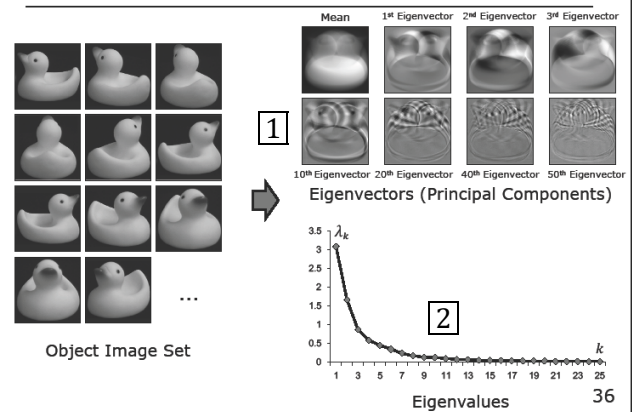
Parametric Appearance Representation

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

35

Dimensionality Reduction

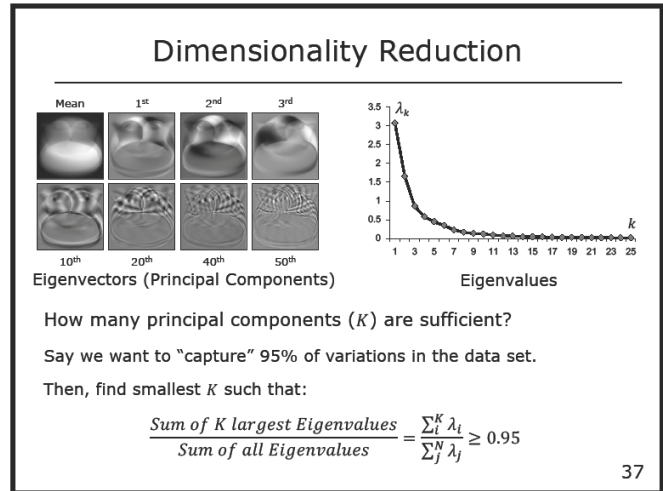


36

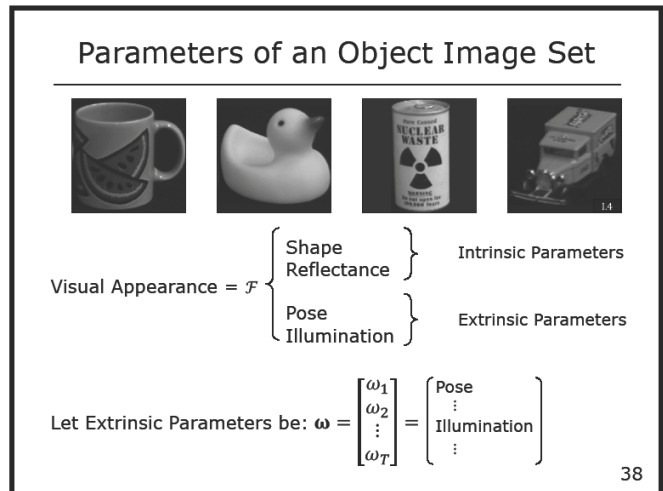
Now that we know how to perform dimensionality reduction using PCA, we are ready to represent the visual appearance of an object. This representation is referred to as the parametric appearance representation. Given an object's image set, such as the one shown here, we first find the mean image. We subtract the mean image from each image in the set, convert the mean-subtracted images into vectors, \mathbf{f}_1 through \mathbf{f}_m , and create the data matrix F . Next, we compute the covariance matrix R and find its eigenvectors [2] and eigenvalues [1].

In [2], the eigenvalues are plotted in descending order of magnitude. We see that they drop quickly in magnitude because of the redundancies in the original image set. The 13th and higher eigenvalues are close to zero, indicating that the corresponding eigenvectors do not contribute significantly to the variance in the distribution of the original image set. It is interesting to note that the higher-order eigenvectors have higher frequency variations within them. The question of interest to us is how many eigenvectors (principal components) do we need to well-represent the original image set.

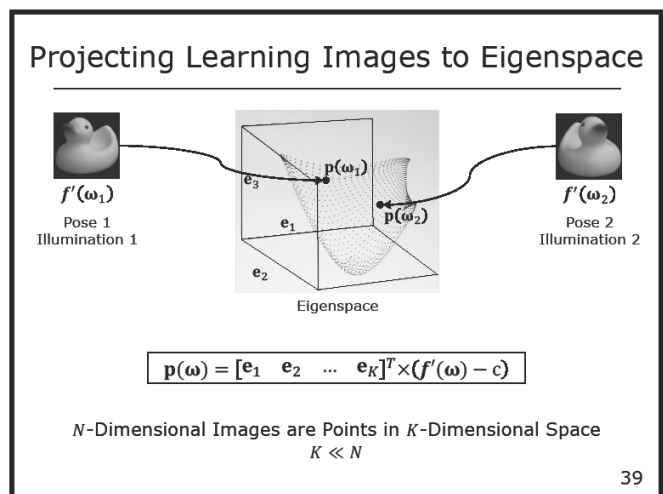
Let us say we want our subspace to capture 95% of the energy in the original image set. What is the dimensionality (K) of the subspace, i.e., the number of eigenvectors, we would need to achieve this? For this, we find the smallest K such that the sum of the K largest eigenvalues divided by the sum of all the eigenvalues is greater or equal to 0.95. In the example shown here, $K = 15$, which means we need a 15-dimensional subspace. If each image in the original set has 200×200 pixels, we have gone from a 40,000-dimensional space down to a 15-dimensional space. This is a significant reduction in dimensionality.



As we discussed earlier, the visual appearance of an object is a function of its intrinsic and extrinsic parameters. The intrinsic parameters of the object include its shape and its surface reflectance. We assume that these intrinsic parameters remain constant across images, meaning that the object is rigid (non-deformable) and its surface properties do not vary. However, we assume that the extrinsic parameters of the object, such as its pose and illumination, could change. We can express these extrinsic parameters as a vector ω . Then, each image in our set is parameterized by ω .

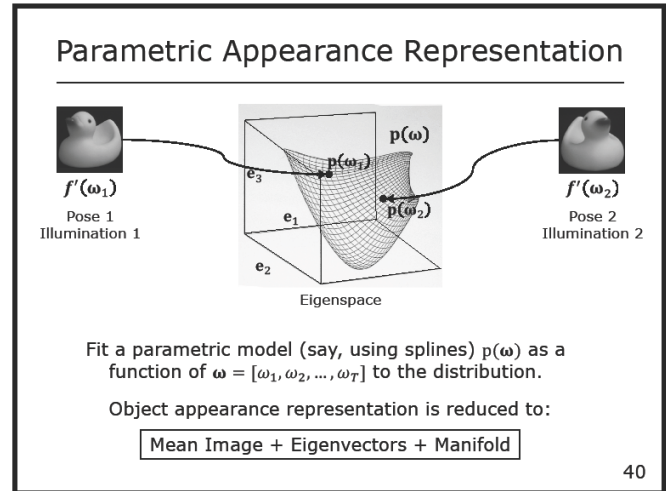


We will project each image in the learning set of the object to the subspace (eigenspace) computed using PCA. Let us consider a 3D eigenspace, represented by the first three eigenvectors e_1 , e_2 , and e_3 , of the duck's image set. Shown here are two images from the set, $f'_1(\omega_1)$ and $f'_2(\omega_2)$, projected after mean subtraction to the eigenspace, to get the points $p(\omega_1)$ and $p(\omega_2)$. When all the images in the learning set are projected in this manner, we get the collection of points seen here. The set of N -dimensional images are now points in a K -dimensional space, where K

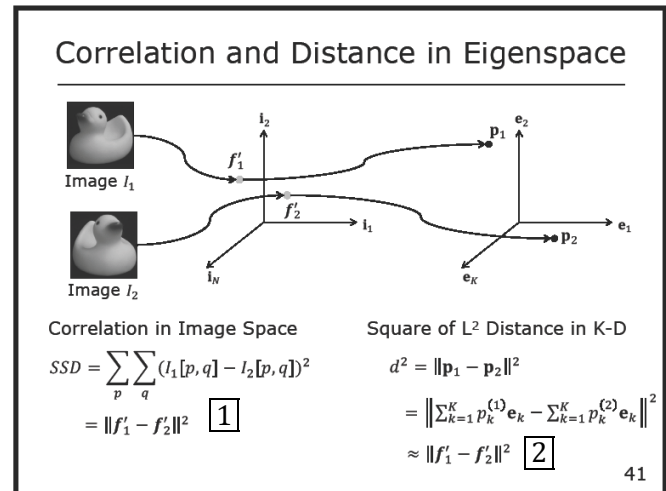


is much smaller than N . Note that two images in the learning set with slightly different pose and illumination, can be expected to be close to each other in eigenspace. By the same argument, the projection of an image of the duck taken with a pose and illumination that lies in between the ones used to capture two consecutive images in the learning set can be expected lie in between the projections of the two images.

Since a novel image of the duck given to us for recognition may have parameters that lie in between the ones used to capture the learning images, we interpolate the projections of the learning image set to obtain a continuous surface in the K -dimensional eigenspace. This surface is referred to as a manifold and it is parameterized by the extrinsic parameters ω . The above interpolation can be done using a low-order surface representation, such as splines. Thus, the complete appearance representation of the object is the mean image (which needs to be subtracted from novel images during recognition), the K eigenvectors that make up the subspace, and the manifold. This is referred to as the parametric appearance representation of the object and is significantly more compact than the original set of learning images of the object.



Let us revisit the naïve method of performing appearance matching by applying template matching. Given two images, I_1 and I_2 , template matching finds the correlation between the two images, which is equivalent to computing their sum of squared differences (SSD). SSD is the L^2 distance between points f'_1 and f'_2 in N -dimensional space, where N is the number of pixels in an image [1]. Now, let us look at the projections p_1 and p_2 of these two images in the K -dimensional eigenspace. Their L^2 distance d in the eigenspace can be expressed in terms of the backward projections of p_1 and p_2 discussed in slide 25. Note that these backward projections are approximations of the original images f'_1 and f'_2 in N -dimensional space. Thus, the L^2 distance d gives us an approximation to the SSD [2]. In fact, there is no K -dimensional space that gives a better approximation to the correlation between the original images f'_1 and f'_2 than the eigenspace.



Appearance Matching

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

42

Appearance Learning (Offline)

Given: M learning images $\{I_1^{(q)}, I_2^{(q)}, \dots, I_M^{(q)}\}$ for each of the Q training objects. $q = \{1, 2, \dots, Q\}$

For each object q , perform steps 1-8:

Step 1: Normalize all images to remove brightness variations.

$$I_m^{(q)} = I_m^{(q)} / \|I_m^{(q)}\|$$

Step 2: Convert image $I_m^{(q)}$ to feature vector $f_m^{(q)}$.

Step 3: Compute the mean feature vector $c^{(q)}$.

Step 4: Subtract from each feature vector the mean vector:

$$f_m^{(q)} = f_m^{(q)} - c^{(q)}$$

43

We now have all the tools necessary to perform appearance matching. We start with appearance learning, which is an offline procedure. Assume that we want to create a system to recognize Q different objects $\{1, 2, \dots, Q\}$. For each one of these Q objects, we will capture M learning images $\{I_1^{(q)}, I_2^{(q)}, \dots, I_M^{(q)}\}$ under different extrinsic parameters. Next, for each object q , we will perform the following steps. We first segment each image and resize it so that the object fits in a square bounding box of fixed dimensions. Next, we normalize each image by its magnitude to remove the effects of the brightness of the illumination, the camera gain, and the camera exposure. Each image is then converted to a vector, $f_m^{(q)}$. The mean $c^{(q)}$ is computed from all the vectors corresponding to the object, and is subtracted from the vectors $f_m^{(q)}$ to get the mean-subtracted vectors $f_m^{(q)}$.

Next, for each object, we construct the data matrix $F^{(q)}$ by stacking up all the mean-subtracted vectors $f_m^{(q)}$. Then, the covariance matrix $R^{(q)}$ is computed as $F^{(q)}F^{(q)T}$, and PCA is applied to $R^{(q)}$ to find the K eigenvectors $\{e_1^{(q)}, e_2^{(q)}, \dots, e_K^{(q)}\}$ that represent the object's eigenspace. As discussed earlier, we know how to determine the dimensionality of the eigenspace, K , by looking at the eigenvalues. We now project all the vectors $f_m^{(q)}$ to points in the eigenspace. We then fit a parametric manifold to these points to get a continuous appearance representation that is parameterized by the extrinsic variables ω , which could include pose and illumination parameters.

Appearance Learning (Offline)

Step 5: Compute the data matrix and covariance matrix.

$$F^{(q)} = [f_1^{(q)} \quad f_2^{(q)} \quad \dots \quad f_M^{(q)}]$$

$$R^{(q)} = F^{(q)}F^{(q)T}$$

Step 6: Compute the K eigenvectors $\{e_1^{(q)}, e_2^{(q)}, \dots, e_K^{(q)}\}$ of $R^{(q)}$ that represent the new orthonormal basis ("eigenspace").

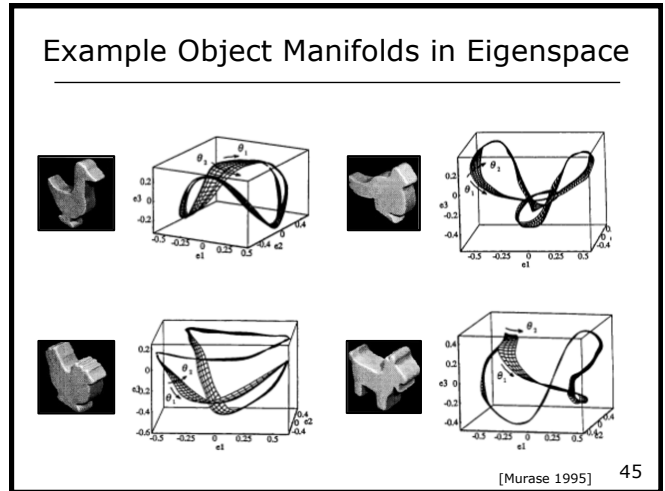
Step 7: Project the learning images to the eigenspace.

$$p_m^{(q)} = [e_1^{(q)} \quad e_2^{(q)} \quad \dots \quad e_K^{(q)}]^T \times f_m^{(q)}$$

Step 8: Fit a parametric manifold to the projected image points as a function of extrinsic variables $\omega = [\omega_1, \omega_2, \dots, \omega_T]$.

44

Shown here are parametric appearance models of four simple objects, parameterized by two extrinsic parameters – a single parameter θ_1 for pose which corresponds to the rotation of the object on a turntable, and a single parameter θ_2 for the direction of a point source (see slide 10). These manifolds are shown in 3D space for visualization purposes, but actually reside in higher-dimensional spaces. Since each object is rotated a full 360 degrees while the learning images are captured, the manifolds are closed.



For recognition, we assume that the object is not occluded in the input image and can be well-segmented and scaled to fit the bounding box we chose for the learning images. We brightness normalize this image I by dividing it by its magnitude $\|I\|$ to get I' , which is then converted to a feature vector f' . For each object q in our database, we perform the following steps. We subtract the mean image $c^{(q)}$ of the object q from the input f' and project it to the object's eigenspace to get a point $p^{(q)}$.

Recognition (Online)

Given: Input image (I) for object recognition.

Step 1: Normalize the image to remove brightness variations:

$$I' = I / \|I\|$$

Step 2: Convert image I' to feature vector f' .

For each object q in the database, perform steps 3-6:

Step 3: Subtract the mean feature vector for object q :

$$f^{(q)} = f' - c^{(q)}$$

Step 4: Project feature vector to eigenspace for object q :

$$p^{(q)} = [e_1^{(q)} \ e_2^{(q)} \ \dots \ e_K^{(q)}]^T \times f^{(q)}$$

46

We then find the distance $d^{(q)}$ between the closest point of the manifold of object q to the projected input point $p^{(q)}$, using a nearest neighbor algorithm. This distance $d^{(q)}$ tells us how close the input point is to the appearance model of the object. We repeat the above process for all the objects in our database, and find the object r for which the distance is minimum. If this minimum distance is less than a threshold, we declare the input image as one of object r . Note that the closest point on the manifold of object r to the input projection also reveals the extrinsic parameters ω , i.e., the pose and illumination of the object in the input image.

Recognition (Online)

Step 5: In the eigenspace for object q , find closest point on the manifold to projected point.

$$\omega^{(q)} = \arg \min_{\omega} \|p^{(q)} - p^{(q)}(\omega)\|$$

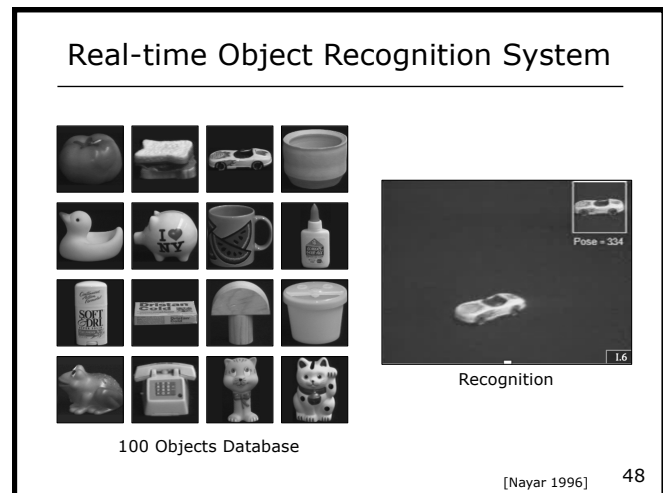
Use a Nearest Neighbor Algorithm for finding closest point.

Step 6: Find the distance $d^{(q)}$ between the projected image point and the closest point on the manifold.

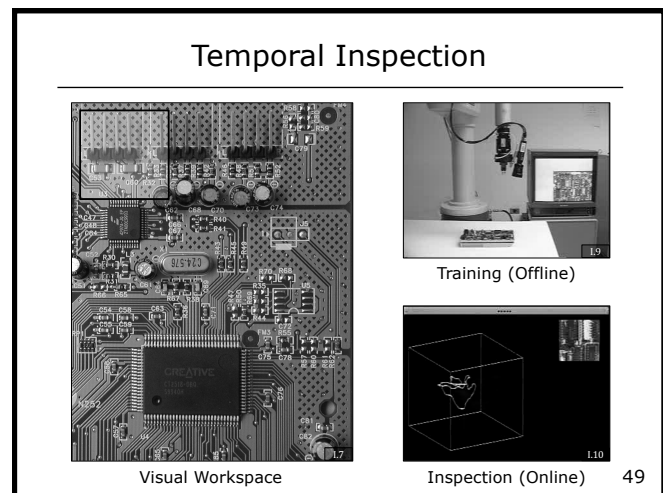
Step 7: Find the object q for which $d^{(q)}$ is minimum.

47

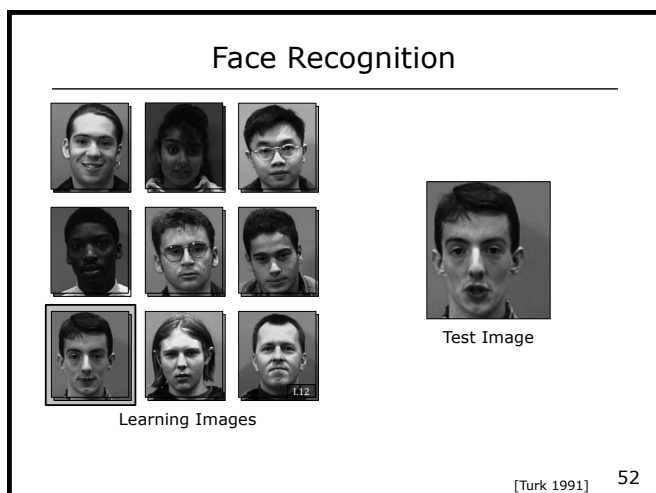
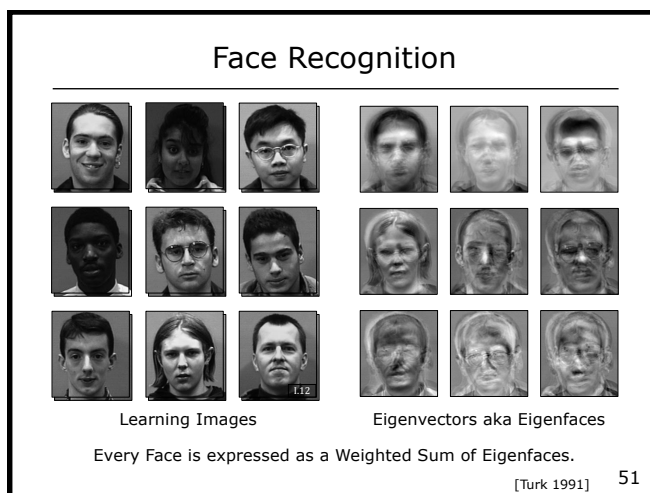
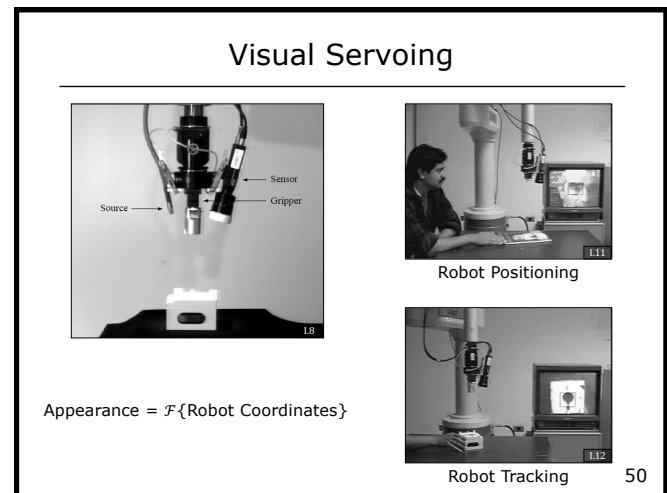
Let us look at a few example applications of appearance matching. Shown here is a recognition system developed about three decades ago with a database of 100 objects. A single parameter was used for the pose, which is rotation about a single axis. The system recognizes the identity and pose of the object presented to it. The online video lecture shows the system working in real time.



Here is the use of appearance matching for temporal inspection. The goal is to quickly scan a manufactured product with a camera mounted on a robot, and detect discrepancies with respect to a stored appearance model. Shown here is the inspection of a printed circuit board, where the task is to detect if all the components of the circuit board are located where they should be. During learning, the robot (top right) traverses a fixed trajectory while the camera captures a video of a model (good) circuit board. The frames of the video are used to compute an appearance model that is parametrized by time during the scanning process. When a new circuit board is presented to the system, the robot traverses the same trajectory and the appearance at each instant of time is compared with the expected appearance. If the appearance deviates from the expected one because, say, a component is missing, the board is deemed to be defective, and is passed along to a human for closer inspection.



Shown here is the use of appearance matching for visual servoing. A robot has a gripper and a camera mounted on its end-effector. We want to perform a particular task, for example a peg-in-hole insertion. However, we do not know the precise location of the hole. During learning, the robot gripper is first positioned exactly where it needs to be with respect to the object with the hole. Then, several displacements with respect to this position are applied to the end-effector and an image is taken for each one. These images are used to compute an appearance model that is parameterized by the displacement parameters. During the insertion task, an image is taken by the camera and this image is compared with the appearance model to determine the displacement of the end-effector from its desired position. This displacement is used to correct the position of the end-effector and then the peg is inserted into the hole. In the bottom right, the object with the hole is being moved. The above method is used by the end-effector to follow the object. When the object stops moving, the peg is inserted in the hole (see the online lecture video).



The use of eigenspaces for face recognition was first demonstrated Turk and Pentland. Shown here on the left of slide 51 are some of the captured images of different people in the database. This entire image set is used to compute the principal components shown on the right, which are often referred to as eigenfaces. As before, each image in the database is projected to a point in the eigenspace. Given a novel face image, it is projected to the eigenspace and the closest point (face in the learning set) determines the identity of the person. As seen in slide 52, the system correctly determines the identity of the person in the novel image on the right as that of the highlighted person (bottom left) in the database. The use of eigenspaces for face recognition was one of the first successful applications of appearance matching.

References and Credits

Shree K. Nayar
Columbia University

Topic: Appearance Matching, Module: Perception
First Principles of Computer Vision

53

References

[Bentley 1975] J. L. Bentley. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975.

[Murase 1995] H. Murase and S. K. Nayar. "Visual Learning and Recognition of 3D Object from Appearance." IJCV, 1995.

[Nayar 1996] S.K. Nayar, S.A. Nene and H. Murase. "Real-Time 100 Object Recognition System." ICRA, 1996.

[Turk 1991] M. A. Turk and A. P. Pentland. "Face Recognition Using Eigenfaces." CVPR 1991.

54

Image Credits

- I.1 <http://www.drububu.com/animation/voxelpusher/index.html> Arjan Westerdiep.
- I.2 www.pointclouds.org/assets/uploads/gsoc14-superquarics.jpg. Licensed under CC BY 3.0.
- I.3 https://en.wikipedia.org/wiki/Constructive_solid_geometry. Licensed under CC BY-SA 3.0.
- I.4 CAVE Lab, Columbia University. Used with permission.
- I.5 CAVE Lab, Columbia University. Used with permission.
- I.6 CAVE Lab, Columbia University. Used with permission.
- I.7 CAVE Lab, Columbia University. Used with permission.
- I.8 CAVE Lab, Columbia University. Used with permission.
- I.9 CAVE Lab, Columbia University. Used with permission.
- I.10 CAVE Lab, Columbia University. Used with permission.
- I.11 CAVE Lab, Columbia University. Used with permission.
- I.12 <http://cswwww.essex.ac.uk/my/allfaces/faces94.zip>. Libor Spacek.

55

Acknowledgements: Thanks to Roshan Kenia, Ayush Sharma and Nikhil Nanda for their help with transcription, editing and proofreading.

References

- [Bentley 1975] Multidimensional binary search trees used for associative searching, Bentley, J. L., Communications of the ACM, 1975.
- [Murase 1995] Visual Learning and Recognition of 3D Object from Appearance, Murase, H. and Nayar, S. K., IJCV, 1995.
- [Nayar 1996] Real-Time 100 Object Recognition System, Nayar, S. K., Nene, S. A., and Murase, H., ICRA, 1996.
- [Turk 1991] Turk, M. A. and Pentland, A. P., Face Recognition Using Eigenfaces, CVPR, 1991.
- [Nayar 2022E] [Image Processing I](#), Nayar, S. K., Monograph FPCV-1-4, First Principles of Computer Vision, Columbia University, New York, March 2022.
- [Nayar 2025B] [Face Detection](#), Nayar, S. K., Monograph FPCV-2-5, First Principles of Computer Vision, Columbia University, New York, February 2025.
- [Nayar 2025K] [Structure from Motion](#), Nayar, S. K., Monograph FPCV-4-4, First Principles of Computer Vision, Columbia University, New York, April 2025.