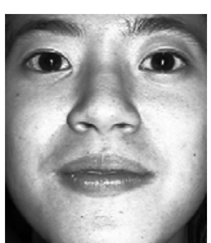
Appearance Based Recognition

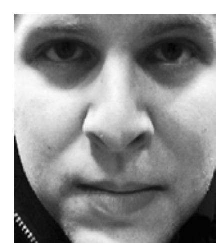
Dr. Gerhard Roth

Recognition (Section 10.4)

- Given an image I, containing a single object and a database of images find the image in the database that is most similar to image I
- One possible way to recognize objects
  - Database has views of same object under different conditions
  - Input image is “close” to one of these database views
- Commonly used in face recognition systems
  - Database has number of faces (standard position)
  - Input image is a single face (standard position)



= ?



Assumptions in appearance recognition

1. Each image contains only a single object.
2. The objects are imaged by a fixed camera under weak perspective.
3. The images are normalized in size: that is the image frame is the minimum rectangle enclosing the largest appearance of this object.
4. The energy of the pixel values is normalized to one: $\sum_{i=1}^{N} \sum_{i=1}^{N} I(i, j)^2 = 1$
5. The object is completely visible and unconcluded in all images.

Comparing Images

- First transform 2D image into a 1D vector
  - N x N image X, becomes an N^2 dimensional vector
  - $x = [X_{11}, X_{12}, \ldots, X_{1N}, X_{21}, \ldots, X_{NN}]^T$
- Now given two images X_1 and X_2, and the two vectors x_1 and x_2 how do we compare them?
- One way, is to find distance between them according to some norm (usually L2)
  - $Dist(x_1, x_2) = || x_1 – x_2 ||$ (just sum of squares of differences)
- If $Dist(x_1, x_2) = 0$ then x_1 and x_2 are identical
- In image processing language
  - $Dist(x_1, x_2)$ is called Euclidian distance (L2 norm)

Comparing Images - Euclidian Distance

- Given a single image y and a database of m images labeled $x_1, x_2, \ldots, x_m$
- Want to find closest image in database to y?
  - Compute Euclidian distance and find smallest result
- How long does this take?
  - Time is proportional to $m * N^2$, where N^2 is number of pixels in the original image, and m number of images in database
  - Takes a long time since N^2 is large and often so is m

Problems with this approach

- May need a very large database
  - Require a different image for different lighting conditions
  - Not much can be done about this issue in a simple way
    - Just hope that enough memory is available
- Often will require a lot of time
  - As in the previous slide to find the best match we need to do convolution against every image in the database
- Can not do much about size of database
  - This problem is intrinsic to the basic approach
- But we can decrease the execution time
  - By using the eigenspace or Principal Components Method
  - Sometimes called the PCA approach

Idea behind the PCA Approach

- There is redundancy in these images
  - Parts of all the images are very similar
  - The faces are in a standard position and all faces are similar
  - We can exploit this redundancy with the PCA approach
- Normally to compare images we need to compare N^2 numbers, but with the PCA approach this is not true
  - We can represent images by k numbers, where $k << N^2$
  - Then to compare images we need only compare k numbers
  - How big is k? It depends on the redundancy of the images
  - If the images are not similar then k is large (close to N^2)
  - The more similar the images the smaller is k
  - For face comparison k is around 100, which is small!

PCA Approach

- To implement the PCA approach you need to
  - Apply the PCA algorithm to all the images in the database
  - Represent each image as a K element vector instead of an N^2 element vector
  - The value of K depends on the redundancy of the images but usually $k << N^2$
- To match a new image with the database
  - Convert the new image into this k element vector form
  - Compare the k element vector to each of the m vectors of k elements in the image database
  - Return the closest vector
- This is the image that is the best match in the sense of Euclidian distance