MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Technical Report No. 1574 March, 1996

Pose-Invariant Face Recognition Using Real and Virtual Views

David Beymer

Abstract

The problem of automatic face recognition is to visually identify a person in an input image. This task is performed by matching the input face against the faces of known people in a database of faces. Most existing work in face recognition has limited the scope of the problem, however, by dealing primarily with frontal views, neutral expressions, and fixed lighting conditions. To help generalize existing face recognition systems, we look at the problem of recognizing faces under a range of viewpoints. In particular, we consider two cases of this problem: (i) many example views are available of each person, and (ii) only one view is available per person, perhaps a driver's license or passport photograph. Ideally, we would like to address these two cases using a simple view-based approach, where a person is represented in the database by using a number of views on the viewing sphere. While the view-based approach is consistent with case (i), for case (ii) we need to augment the single real view of each person with synthetic views from other viewpoints, views we call "virtual views". Virtual views are generated using prior knowledge of face rotation, knowledge that is "learned" from images of prototype faces. This prior knowledge is used to effectively rotate in depth the single real view available of each person. In this thesis, I present the view-based face recognizer, techniques for synthesizing virtual views, and experimental results using real and virtual views in the recognizer.

This thesis describes research done at the Artificial Intelligence Laboratory and within the Center for Biological and Computational Learning. This research is sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879; by a grant from the National Science Foundation under contract ASC-9217041. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038. The author is supported by a Howard Hughes Doctoral Fellowship from the Hughes Aircraft Company.

Pose-Invariant Face Recognition Using Real and Virtual Views

by

David James Beymer

B.S., Computer Science
University of California, Berkeley
(1987)

S.M., Computer Science
Massachusetts Institute of Technology
(1989)

Revised version of a thesis submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the
Massachusetts Institute of Technology
September 1995

Pose-Invariant Face Recognition Using Real and Virtual Views

by

David James Beymer

Abstract

The problem of automatic face recognition is to visually identify a person in an input image. This task is performed by matching the input face against the faces of known people in a database of faces. Most existing work in face recognition has limited the scope of the problem, however, by dealing primarily with frontal views, neutral expressions, and fixed lighting conditions. To help generalize existing face recognition systems, we look at the problem of recognizing faces under a range of viewpoints. In particular, we consider two cases of this problem: (i) many example views are available of each person, and (ii) only one view is available per person, perhaps a driver's license or passport photograph. Ideally, we would like to address these two cases using a simple view-based approach, where a person is represented in the database by using a number of views on the viewing sphere. While the view-based approach is consistent with case (i), for case (ii) we need to augment the single real view of each person with synthetic views from other viewpoints, views we call "virtual views". Virtual views are generated using prior knowledge of face rotation, knowledge that is "learned" from images of prototype faces. This prior knowledge is used to effectively rotate in depth the single real view available of each person. In this thesis, I present the view-based face recognizer, techniques for synthesizing virtual views, and experimental results using real and virtual views in the recognizer.

Thesis Supervisor: Dr. Tomaso Poggio
Title: Uncas and Helen Whitaker Professor, Dept. of Brain and Cognitive Sciences

Acknowledgments

I would like to thank my thesis advisor, Tommy Poggio, for being a constant source of ideas and inspiration, both for my thesis topic and for related lines of work such as applying example-based learning to image analysis and synthesis. I owe special thanks to Amnon Shashua for suggestions early on in my thesis work, especially for demonstrating how well optical flow can be used for geometrically registering two face images. I would also like to thank Professors Eric Grimson and Rod Brooks for being on my thesis committee.

There are many students at the MIT AI Lab that I would like to thank. Jim Hutchinson, one of my officemates, gave me a small education in neural networks by answering a steady stream of questions from me on the subject. Brian Subirana, another officemate, always loved to analyze and evaluate computer vision algorithms. Jose Robles was very helpful in polishing my thesis defense. Kah-Kay Sung and I discussed machine learning and techniques for automatically detecting faces. Other students in the computer vision group at the AI Lab that I had interesting conversations with include Tao Alter and Gideon Stein. I also enjoyed interacting with Steve Lines, Raquel Romano, and Tony Ezzat, fellow students in Tommy's group working on faces.

I would like to thank a couple people at the Center for Biological and Computational Learning. Federico Girosi proofread a couple of my papers and answered some of my general questions about machine learning. Marney Smyth edited a couple of videotapes that demonstrated my thesis work and related applications.

While performing my thesis research, I was supported by the Hughes Aircraft Company, and I gratefully acknowledge their support. Further, I would like to thank Mike Daily at Hughes Research Laboratories for allowing me to continue my thesis work while at Hughes in the summers of 1992 and 1993. Also at Hughes, Trish Keaton also worked on a face recognition project, and I enjoyed discussing the topic with her.

I owe a big thanks to everyone at the MIT AI Lab and Hughes Research Laboratories who volunteered to be in my face database.

Finally, I am grateful to my parents for their constant moral support.

This thesis describes research done at the Artificial Intelligence Laboratory and within the Center for Biological and Computational Learning. This research is sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879; by a grant from the National Science Foundation under contract ASC-9217041. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038. The author is supported by a Howard Hughes Doctoral Fellowship from the Hughes Aircraft Company.

Contents

Chapter 1

Introduction

Using our sense of vision, we are able to effortlessly and instantaneously recognize among thousands of objects stored in our memory. This ability enables us to perform such necessary but mundane tasks as recognize our family pet, avoid cars racing down city streets, and find our coffee mug at work. Looking beyond the human ability, science fiction is full of futuristic androids and robots that can, among other things, visually recognize objects in their environment. But can we actually endow machines with this useful skill? This is the goal of researchers in the field of object recognition, a field guided by the belief that this skill is an important link in making machines exhibit intelligence.

Automatic techniques for object recognition attempt to identify instances of known objects in a digitized image of a scene. Objects are made "known" to the computer in a preliminary stage that may work, for example, by training the machine on example 2D images of objects or by developing a 3D CAD object model. After building up a database of known objects, the computer recognizes instances of these objects in 2D images by matching object models to regions of the 2D image actually containing objects. The pose of objects in the scene can be calculated from the results of the matching process.

Object recognition is an interesting and challenging problem for researchers because of the possibility of building a machine that can interact intelligently with its environment. Automatic object recognition could help provide the "eyes" for a computer, enabling commercial applications like industrial parts inspection and computer-assisted surgery, the latter of which is just beginning to emerge. Existing systems for object recognition, however, fall far short of human abilities. The ease in which the human brain can recognize objects often leads us to underestimate the difficulty of the overall problem.

Why is object recognition a difficult problem? The biggest problem is that object image appearance depends on pose and lighting conditions. That is, a single 3D object can take on one of a multitude of appearances when projected into the 2D image.

Another problem is scene clutter. If the image scene contains many objects, it may be difficult to isolate regions of the image that contain a single object. Furthermore, if objects are allowed to partially occlude each other, then only incomplete information is available to the recognizer. Finally, imaging devices such as cameras and human eyes are imperfect: real lenses tend to blur the image, and the light measurement process invariably adds noise to the image.

While different approaches exist to the problem of object recognition, a simple and direct one is the *view-based* approach. In this approach, variation in object appearance due to changes in pose and lighting are represented simply by storing many different example 2D views of the object. When attempting to recognize a new input view, the recognizer simply tries to match the input against an example view that is sufficiently close in pose and lighting. Since pose-lighting parameter space is multidimensional, populating this space densely enough with example views could require a huge number of examples. Thus, one important issue in the view-based approach is how many example views are necessary for good recognition performance.

The goal of this thesis is to explore the view-based approach in a real application domain, face recognition, under different extremes in terms of the number of available example views per person. We will examine two cases:

1. many example views are available per person, and

2. only one example view is available.

The application problem itself, chosen to be a good candidate for the view-based approach, will be pose-invariant face recognition. By "pose-invariant", we mean that the pose of the input view is free to vary within a certain range. Since the input pose is allowed to vary, applying the view-based approach naturally requires many example views of each person, views in this case taken from different poses. While this is compatible with the first case above, what about the second scenario where we only have one example view? Assuming that the single view is from a fixed pose r_{std}, will the face recognizer be successful for input views at poses distant from pose r_{std}? At first this may seem to be a problem with applying the view-based approach.

To assist the view-based approach in the single example view case, we try to exploit properties of the generic class of object being recognized to generate additional *virtual* examples. That is, even though we have a single view per object, prior knowledge about the general class of objects may enable us to generate additional examples. In the case of pose-invariant face recognition, the generic object class is the human face, and virtual views are views as seen from poses different from that of the single real example. The prior knowledge of faces needed to generate these virtual views will be of face rotation, representing how a face transforms when rotated. After presenting our technique for generating virtual views, their usefulness will be evaluated in a view-based pose-invariant face recognizer.

In this thesis we make two main contributions. First, our view-based face recognition work is the first to systematically explore the problem of pose-invariant face recognition. As we will discuss in the prior work section, most existing work in face recognition covers a small range of poses and uses at most a few example views per person. In the case when multiple real views are available, we show that a simple view-based approach yields good recognition rates. Second, in situations where only one view per person is available, we show that the view-based approach is still viable by developing techniques for synthesizing virtual views.

1.1 Object Recognition

The goal of object recognition is to automatically identify instances of objects that the computer has been trained or programmed to recognize. That is, given prior knowledge of a set of known objects and a digitized image of a scene to analyze, object recognition systems attempt to locate, recognize, and estimate the orientation of objects in the scene. For example, consider the problem of recognizing a telephone in an input image of an office desktop scene, an image perhaps cluttered with other common desktop objects such as pens, books, etc. The object recognizer tries to match a model of the telephone, perhaps a 3D CAD model, to the region of the image containing the telephone.

There are three major approaches to object recognition, an invariant features approach, a model-based approach, and a view-based approach. The basic difference among the three approaches is how they deal with variation in object appearance across different image parameters such as pose and lighting. The invariant features approach finds object features that do not change as image parameters change, sidestepping the problem of actually representing changing object appearance. The model-based approach, the most popular approach, uses 3D object models to predict appearance under different image parameters. Finally, the view-based approach handles variation in appearance by simply storing many 2D views of the object.

Before going into more detail, let us introduce some relevant terms:

representation To manipulate object models mathematically or inside a computer, we must represent them in terms of primitive elements. A very wide variety of representations have been explored. Probably the simplest one is the template-based representation, where 2D images, or templates, of the object are used to represent appearance. In this pictorial representation, the pixels of the template can be interpreted as the primitive elements. Going beyond the original grey levels, some representations transform the image using the KL or Fourier transforms, and some pass the image through filters like Gabor filters. Transform coefficients and filter outputs are the primitives here. Moving to more symbolic primitives, some systems use the geometrical configuration of point, line, or contour features. Even more abstract are the volumetric constructs of superquadrics and generalized cylinders. The latter symbolic primitives are often defined in 3D and used in the model-based approach. Representations using templates or transformed/filtered versions of the image are inherently 2D and are often paired with the view-based approach.

correspondence When recognizing objects in an input image, one of the first steps is to locate the representation primitives in the image, a process known as feature detection. Correspondence refers to the pairing of model primitives with image features. A particular correspondence, or pairing, determines a matching of the model and input, and is often used to specify a geometrical transformation between the two.

The descriptive power of the model primitives and image features has an impact on the number of admissible correspondences. For example, if the primitives and features are rich and detailed, then the number of potential matchings may be very small, or perhaps a unique correspondence may exist. On the other hand, if features are generic, such as point features, then a large number of potential correspondences may be explored by the recognizer.

pose Pose refers to the rotation and translation parameters that describe the orientation and position of the image object with respect to the model. For 3D objects, one also has to model the projection process from 3D to a 2D image, so the overall mapping goes from a 3D object-centered coordinate system to 2D image pixel coordinates. Consider the simple *scaled orthographic* projection model for transforming a 3D object-centered point $\mathbf{p}$ to its corresponding 2D image location $\mathbf{p}'$:

$$\mathbf{p}' = s\,P R\,\mathbf{p} + \mathbf{t}. \tag{1.1}$$

This transformation has six degrees of freedom: three rotations, one scale, and two translations. First, the point $\mathbf{p}$ is transformed by a standard 3x3 rotation matrix R. Then it is projected using a 2x3 orthographic projection matrix P, which simply drops the z coordinate. The overall scale of the object is then determined by a scale factor s and then the object is translated in 2D by an offset vector $\mathbf{t}$. This is an approximation of standard perspective projection, an approximation that works well when the extent of the object in depth is small when compared to the distance between the object and camera.

In the case of 2D object models, a less complicated pose model is required, a model that needs only to map the image plane back onto itself. The mathematical form is similar to equation (1.1)

$$\mathbf{p}' = s\,R\,\mathbf{p} + \mathbf{t}, \tag{1.2}$$

except that $\mathbf{p}$ is now 2D and the rotation matrix R is a simple 2x2 rotation matrix that operates only in the image plane. There are four degrees of freedom: one image-plane rotation, one scale, and two translation parameters.

Object recognition systems are usually divided into two stages, a model acquisition or training stage and a matching stage. The model acquisition stage is an off-line step that builds an object representation for later use during the actual on-line matching stage. For the view-based approach, this stage really builds a set of representations. In the on-line matching stage, given an input image to process, the object recognizer finds a way to match a model representation with a subregion of the image containing the object. The matching stage involves detecting image features and solving for correspondence and pose either explicitly or implicitly. Let us now review how these steps work in the three previously mentioned approaches to object recognition.

1.1.1 Invariant features approach

In the invariant features approach, the key is to find an object representation that does not vary as image parameters such as pose and lighting are varied. The matching process in the recognizer is then quite simple: compute the invariant representation in the input image and compare against the stored model representation. This approach is possible when one can find features where image parameters are decoupled from object identity. When such features exist, this is clearly preferable to the model-based and view-based approaches, where imaging parameters are factored into and thus complicate the recognition process.

A simple example of the invariant features approach uses the so-called cross ratio. Consider four collinear points A, B, C, and D and let AB denote the 2D distance between A and B in the image. Then the ratio $\frac{AC}{BC} \div \frac{AD}{BD}$ is invariant with respect to pose. Using methods from invariant theory, Forsyth, *et al.* [54] develop geometric invariants of a similar flavor for 3D planar objects using contour and point features. The color of an object is another invariant feature. Swain and Ballard [126] use a histogram of image color values to perform indexing, roughly defined as the culling of object models prior to more detailed matching. In an effort to detect faces under varying lighting conditions, Sinha [122] first averages brightness values over a set of facial regions including the eyes, nose, cheeks, mouth, chin, etc. He shows empirically that pairs of regions exist where the average brightness of one region is consistently greater than that of another (e.g. cheeks greater than eye region), and he represents a face as a collection of such relative magnitude tests.

However enticing this approach may appear, it does have limitations when applied to more complicated 3D objects. For instance, it has been shown (see Clemens and Jacobs [41], Burns, Weiss, and Riseman [30]) that for a 3D object consisting of an arbitrary set of points there are no geometric invariants. That is, when objects are modeled as a 3D cloud of points – as contrasted with the special case of planar objects before – there are no simple invariant functions such as the cross ratio.

1.1.2 Model-based approach

The model-based approach to object recognition uses geometric models, usually 3D ones, to predict the appearance of the modeled object under arbitrary pose. Most of the work in model-based recognition has been applied to rigid, man-made objects such as machined parts, staplers, computer mice, etc. The popular representation in the model-based approach are intensity edges or features derived from edges such as corners, junctions, or edge normals. By focusing attention on the regions of intensity discontinuities, the representation is fairly invariant to lighting conditions, so model-based systems address the problem of varying pose.

Consider the typical case of a 3D object model and a 2D image. During the model acquisition stage, a 3D model of geometric primitives is constructed, usually built from point features or edge fragments. This model is defined in an object-centered coordinate system. Given a particular pose, equation (1.1) can be used to project the primitives into the image plane, where they can be compared with image features.

At recognition time, the first step is to detect point and edge features, the instances of model primitives. The matching procedure then solves for a correspondence and pose that overlays the projected model onto the set of image features belonging to the model. While many matching strategies exist, consider the popular alignment scheme of Huttenlocher and Ullman [70]. Recall from our definition of terms how correspondence and pose are linked. By specifying enough feature correspondences, one determines a potential pose of the object. Thus, one way of exploring the possible model-to-image transformations is to enumerate all possible feature correspondences, a process that unfortunately grows exponentially with respect to the number of features. The key observation of Ullman and Huttenlocher is to use a *minimal* number of features in forming correspondence. They show that for the 3D problem, only three point correspondences are sufficient for determining pose. That is, the recognizer only has to enumerate all correspondences of three model features to three image features to examine all possible poses. The recognizer then enters a verification phase where it examines the pose specified by each correspondence. Here the algorithm returns to the full detail of the model representation, using the hypothesized pose to project all features into the image plane. Matching is then performed in 2D between the projected model features and the image features. Related systems include the interpretation tree method of Grimson and Lozano-Pérez [62], the local feature focus method of Bolles and Cain [22], and the SCERPO system by Lowe [89].

While the previous approaches organize the matching process as a search over correspondences, another way to view the search is in pose space. From this viewpoint, the matcher tries to find a pose that brings a large number of projected model features into correspondence with image features. This is the basis of the *Hough transform* technique (see Ballard and Brown [10], Chapter 4 for a good introduction). A mathematically elegant way of relating correspondence space to pose space for the problem of *bounded error* recognition was formulated by Baird [9]. The constraint of bounded error means that projected model features fall within a certain bounded function centered around the image features. This idea was further developed theoretically and algorithmically by Cass [36] and Breuel [24].

1.1.3 View-based approach

As previously discussed, the view-based approach to object recognition models 3D object appearance simply by storing many example 2D views of the object. The views may be from a variety of poses, lighting conditions, and for the case of faces, expressions – basically whatever sources of variability that one wishes to handle in the input views. Compared with the 3D model-based approach, it is often said that the view-based approach trades memory for computation. That is, storing a number of views takes more memory than a single 3D object model, but typically the matching task is less computationally expensive since it is performed in 2D rather than 3D. If we consider for a moment this trade-off in terms of the human brain, the view-based approach is often considered to be a more biologically plausible one than the 3D model-based approach since our brains have massive amounts of memory and rather slow computational speed. In machine applications of object recognition, the cheapness of digital memory may make the view-based approach practical in real implementations.

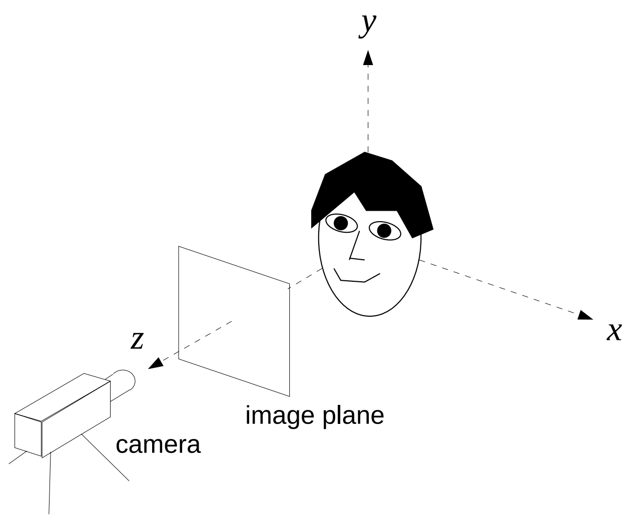


Figure 1-1: In scaled orthographic projection, the object in the 3D object-centered coordinate system (x, y, z) is rotated in 3D, translated in the x-y plane, orthographically projected along the z-axis, and then scaled in the image plane. The view-based approach samples object views at different pairs of rotation angles about the x- and y-axes.

Application to pose problem

Consider applying the view-based approach to the problem of recognizing a 3D object under varying pose. Following the prescribed approach, we need to take example views of the object to capture changes in appearance as pose parameters vary. Referring back to our discussion of scaled orthographic projection, there are six degrees of freedom to pose, two translations, a scale factor, and three rotations. Do we actually need to worry about all six of these dimensions? Fortunately, four of the parameters, the two translations, scale, and image-plane rotation (rotation about the z-axis), do not change image appearance in a fundamental way. For example, given an image of an object, we can model changes in 2D translation simply by translating the image in 2D. In general, we can model change in appearance for those four parameters simply by applying a planar transform to a single image. Thus, there is no need to take example views along those pose dimensions.

To model object appearance with respect to pose, then, we are left with the pose rotations "in depth", or the rotations about the x- and y-axes (see Fig. 1-1). These pose parameters change object appearance in a manner that cannot be handled using a simple planar transform. If one imagines the object as enclosed in a sphere and rotates the camera around this sphere instead of keeping it fixed on the z-axis, these two rotational parameters describe where on the sphere one is viewing the object; hence this imaginary sphere is often called the *viewing sphere*. Thus, the model acquisition phase consists of obtaining a set of object views that sample poses from the viewing sphere.

During the recognition of a new input view, the view-based approach typically cycles through the example views, comparing each example view to the input using a 2D matching algorithm. By going though the different example views, the rotation angles in depth are examined. The remaining pose parameters, the two translations, scale, and image-plane rotation, are typically handled by the 2D matching algorithm. The 2D matching step can be seen as a special 2D case of the model-based matching approach, consisting of detecting features and then searching over correspondence or pose space. The amount of search is determined by the descriptiveness of the features. If the matching procedure is driven using generic features such as points and lines, the matcher will have to search over a set of potential model-image correspondences. On the other hand, if the features are distinctive, such as the eyes of a face, then aligning the model and input views in 2D is straightforward. In this latter case image features are used to geometrically normalize the input for the 2D aspects of pose.

Prior work

One of the key components of a view-based system is the manner in which the example views are chosen. Basically, how is the viewing sphere sampled? One of the simpler techniques is to sample the entire sphere or portions of it at regular intervals. Goad [60] samples 216 points on the viewing sphere by imposing a 6x6 grid on the 6 faces of a unit cube and then radially projecting the cube onto the viewing sphere. The object domain of Goad's system is industrial parts and the example views are represented using 2D edges. In the view-based system of Breuel [23], toy models of two airplanes are represented by sampling the upper half of the viewing sphere; 32 images were used for one plane and 21 for another. A 2D edge-based representation is again used, and the 2D matching algorithm employed is Breuel's RAST algorithm, an algorithm that searches pose space using an adaptive subdivision technique.

One of the problems of the regular sampling approach is determining an appropriate number of examples. Sampling the viewing sphere too finely may result in high storage costs, while sampling too coarsely may accidentally miss a view with a unique configuration of features. The technique of *aspect graphs* attempts to find a compact set of views that effectively covers all views of the object.

As introduced by Koenderink and van Doorn [78], an *aspect* is roughly defined as an object view whose features are stable with respect to pose perturbations. An aspect carves out a patch of qualitatively similar views from the viewing sphere. If the pose is rotated enough, then the set of image features change, creating a new aspect, and a *visual event* is said to have occurred. In an aspect graph, the nodes are a collection of distinct aspects of an object, and there is an edge between two nodes if the aspects are connected by a visual event. The theory behind aspect graphs is developed using 3D solid shape and ignores object albedo, so image features are edge-based and thus visual events refer to changes that occur in the configuration of edges and junctions.

The focus of aspect graphs is more on creating the set of aspects rather than actually using it for recognition, although recognition is the assumed eventual goal. Chakravarty and Freeman [37], working within the same framework but calling aspects *characteristic views*, develop a set of rules for manually generating a set of characteristic views from a 3D object model. Later work has focused on automatically computing the aspect graph. Korn and Dyer [81] and Ikeuchi and Kanade [71] compute the aspect graph by first enumerating a large number of views and then clustering similar views into aspects. However, the clustering approach is limited by the fineness of the initial sampling; an aspect with small support on the viewing sphere could still be missed. Exact methods for computing the aspect graph basically work by mapping visual events onto the viewing sphere, partitioning it into aspects. Stewman and Bowyer [123] and Gigus and Malik [58] have shown how to do this with polyhedral objects; Ponce and Kriegman [111] with curved objects constructed from parametric surfaces. It should be added that the systems in [37] and [71] also develop recognition strategies using their aspect graph representation.

The *linear combination* approach to object recognition (see Ullman and Basri [130]) is related to the view-based approach in that a set of 2D views is collected in the model acquisition phase. However, in linear combinations, instead of storing these views as isolated examples, the example views are interpolated to generate new views of the object. That is, Ullman and Basri show that any 2D view of an object can be written as a linear combination of example 2D views. Only two example views are needed for objects with "sharp edges" such as polyhedral objects (Poggio [105] also showed this result), with more views being required if smooth surfaces or articulated objects are allowed. The 2D view representation is shape-based and thus relies on point and contour features. The technique requires these features to be in correspondence across both the example views and new views being recognized, a correspondence that Ullman and Basri assume is externally supplied. The analysis assumes an orthographic projection model; see Shashua [121] for more recent work on extending the linear combinations approach to the perspective case.

Overall, the linear combinations technique shows that a set of 2D views is equivalent to having 3D structure. In this case 3D models can be bypassed by taking an appropriate linear combination of 2D views. This is similar to binocular stereo and structure-from-motion algorithms, where multiple 2D views of an object are used to compute 3D structure.

Example-based learning techniques, when applied to the problem of object recognition, can be read as a view-based approach to the problem. Compared to the previous approaches, the basic idea is to have the computer *learn* how to perform the recognition task rather than having to explicitly code a recognition strategy. This approach has recently been explored by Poggio and Edelman [107] and Brunelli and Poggio [26]. Given a set of example 2D views of an object, example-based learning techniques try to construct a mapping from the space of 2D views to some property of the object, where that property may be an indicator variable (1 if the view is from the object, 0 otherwise) or an estimate of 3D object pose. Being able to learn mappings of this type from a sparse set of examples is possible because these mappings are typically smooth (i.e. the appearance of an object changes slowly when rotated). Similar to the linear combinations approach, feature correspondence among the different views is assumed.

In this technique, first a mathematical form for the mapping is chosen, along with an associated architecture that implements the mapping in terms of a network of simple interconnected nodes. Next, in the network training procedure, which is basically the model acquisition phase, example pairs $(\mathbf{x}, y)$ of input views $\mathbf{x}$ and desired output y are presented to the network; the training procedure adjusts the parameters of the network so the network "learns" a function f such that $f(\mathbf{x}) \approx y$. After training, at recognition time, a new view $\mathbf{x}$ is presented to the network and the output y is computed by the network. When network output is taken to be an indicator variable for the object, a frequent case, the network "recognizes" the object.

Both [107] and [26] use Radial Basis Functions to approximate f (see Poggio and Girosi [109]), along with a simple accompanying three layer network. The object domain is wire-frame "paperclip" objects, and inputs $\mathbf{x}$ are taken to be the (x, y) coordinates of wire intersections. In the recognition experiments, a relatively small number of objects is used (5 or 10) and the number of example views used to train the network is in the tens (Poggio and Edelman [107] suggest 80-100).

In another learning approach, Murase and Nayar [98] have explored a matching method based on the 2D appearance of an object. Instead of focusing on object shape by looking at edge-based and point features, they represent object views by the grey level appearance of the object, which includes factors like illumination and object reflectance as well as shape. As in the view-based approach, a large set of views per object are collected that sample the viewing sphere and different lighting directions. But instead of storing all these views, they are compressed using principal components analysis. Object views are represented by projecting them onto a small number of the principal components, forming an "eigenspace" representation. Murase and Nayar then represent an object by fitting a parametric surface to the projection of its example views in this eigenspace. Given a new image to recognize, it is projected into the eigenspace and compared with the hypersurfaces of the various modeled objects. This technique as been tested with large numbers of example views (450) and testing views (270) per object. Overall, the compression from principal components allows an approximation to correlation with the entire original data set but at only a fraction of the cost.

Overall, one of the important components in the view-based approach is the set of example views used to sample the viewing sphere. Determining a small but complete set of example views is the primary problem of aspect graphs. Systems that use more systematic and dense sampling of the viewing sphere, such as the eigenspace approach, spend much effort in collecting the example views. The number of example views required by the view-based approach can be addressed mathematically using error analysis (Breuel [23]). Poggio and Edelman [107] estimate that around 80-100 views may be sufficient for a network approach.

As previously mentioned, in this thesis we focus on two extreme cases in terms of the number of available views. In the first case, where many views per object are available, our approach uses a regular sampling of a portion of the viewing sphere. In the second case, however, only one view is available, so we develop a technique for generating additional views of the objects, or "virtual" views. We now examine virtual views in more detail.

1.2 Exploiting Prior Knowledge of Object Classes

A general problem for the view-based approach to object recognition and more generally, example-based learning, is that not enough examples may be available for a particular problem. A lack of examples may seriously limit the scope of inputs that a learning network can properly generalize over. In the case of view-based face recognition, having just one view per person, say at pose r_{std}, means that we would expect recognition performance to drop as input pose differs more and more from r_{std}. Being attracted by the simplicity of the view-based approach, though, we would like to know if it is possible to address this problem within the view-based framework. Considering that learning systems and object recognition systems are usually applied to specific classes of objects, such as manufactured parts, faces, and characters (OCR), can we use knowledge of this class of objects to leverage the example set? In this thesis we explore using prior knowledge of the object class to expand the set of examples by generating *virtual* examples, or virtual views.

1.2.1 Virtual views using 3D models and symmetry

Given a single view of an object at pose r_{std}, consider the problem of generating a rotated virtual view of the object, say a view that has been rotated about the y-axis by 15 degrees. To perform this task, the prior knowledge must say something about the 3D shape of the object. Probably the most straightforward way to represent knowledge of 3D shape is with an explicit 3D model. This approach works well when the class of objects has a common, generic 3D model, which is certainly true for an object class like faces. Researchers in computer graphics, low-bandwidth teleconferencing, and performance-driven animation have already exploited 3D models in this way to generate new views of a face when provided with just one view. First, the single real view is texture mapped onto the 3D model, then the 3D model is rotated to the new desired pose, and finally the rotated 3D model is projected onto the image plane. This technique will be discussed in more detail in the next chapter on prior work.

Knowledge about symmetries of 3D shape can also be used to generate rotated virtual views. In an early demonstration of the idea of virtual views, Poggio and Vetter [110] showed that for bilaterally symmetric objects, knowledge of symmetry allows one to synthesize a virtual view from a single view. The virtual view is simply the mirror reflection of the real view. Combining this with earlier ideas about linear combinations leads to a surprising result. According to linear combination of views, two views are sufficient for recognition. Since we can get a mirror view of a bilaterally symmetric object essentially for free, it should be possible to recognize a symmetric object using just *one* example view. Thus, while one view alone is not sufficient for recognition, when paired with prior knowledge of symmetry it becomes possible.

Before applying this technique for recognition from one view, there are some theoretical and practical issues to consider. First, the virtual view is degenerate – equal to the original real view – when object pose is such that the plane of symmetry is perpendicular to the image plane. For faces this occurs in frontal pose, so for virtual views based on symmetry we need to use an "off-center" view of the face. Second, the linear combination of views result assumes correspondence between views, and unfortunately the symmetry result does not supply correspondence. Thus, difficulty in finding correspondence will limit the ability to apply the virtual view to recognition. Unfortunately these two considerations play off one another in a negative way: to the extent that one avoids degeneracy by choosing an off-center view, one makes the correspondence problem more difficult since the poses of the two views will be further apart.

1.2.2 An example-based approach to virtual views

In this thesis we consider using an *example-based* approach to representing prior knowledge of 3D object shape. By example-based, we mean that prior knowledge will be represented by a set of 2D views of prototype objects, where the prototype objects are representative of the variation seen across the object class. For each prototype object we assume that there are many available views, as opposed to non-prototype objects for which just one view is available. In particular, let the single view that is available for non-prototype objects be at a standard view r_{std}, and let the $\{r_i\}_{i=1}^n$ be a set of n poses of desired virtual views. The example-based approaches we explore will then require $(n+1)$ views per prototype at poses r_{std} and $\{r_i\}_{i=1}^n$. Given a view of a non-prototype, or novel, object at pose r_{std}, the example-based approach will generate a view at one of the desired virtual poses.

Two example-based approaches will be explored in this thesis for generating virtual views:

1. *Parallel deformation.* Using just one prototype object, measure the 2D deformation of object features going from the standard to virtual view. Then map this 2D deformation onto the novel object and use the deformation to distort, or warp, the novel image from the standard pose to the virtual one. The technique has been explored previously by Brunelli and Poggio [108] within the context of an "example-based" approach to computer graphics and by researchers in performance-driven animation (Williams [136][137], Patterson, Litwinowicz, and Greene [102]).

2. *Linear classes.* Using multiple prototype objects, first write the novel object as a linear combination of prototypes at the standard pose, yielding a set of linear prototype coefficients. Then synthesize the novel object at the virtual pose by taking the linear combination of prototype objects at the virtual pose using the same set of coefficients. Using this approach, as discussed in Poggio [106] and Poggio and Vetter [110], it is possible to "learn" a direct mapping from standard pose to a particular virtual pose.

Why should we use the example-based approach over the 3D model and symmetry approaches described above? Symmetry can only provide one additional view, which will not be sufficient for our application to pose-invariant face recognition, as we will see experimentally. The case of 3D models breaks down into two subcases. Some systems for the 3D modeling of faces use person-specific 3D models taken with an active sensor such as the Cyberware scanner. These types of individual 3D models fall outside the scope of our recognition-from-one-view scenario, for the system has much more than one 2D view – it can generate any 2D view it pleases. The second case of 3D models is a generic 3D face model. The main problem of using generic 3D models is adjusting the 3D model so that it matches the 2D novel object view. This may work when a couple of different novel object views are available, such as with a pair of frontal and side views of a face (see Wallace [6]).

1.2.3 Related work

Hints have been proposed by Abu-Mostafa [1][2] as a method for incorporating prior knowledge into the learning-from-examples framework. In the standard learning-from-examples framework, the goal is to learn a function f from a set of hypothesized functions G, where one can think of G as a class of functions implemented by a particular neural network architecture. A learning procedure, such as backpropagation, chooses a particular function $g \in G$ based on a set of input-output examples of f, $(\mathbf{x}_i, f(\mathbf{x}_i))_{i=1}^{n}$, where $\mathbf{x}_i$ is a member of the input space X. A hint is any prior knowledge that reduces the size of G, thus making the learning task easier. Abu-Mostafa introduces a couple of types of hints and discusses a method for incorporating hints into the learning procedure.

For example, one type of hint is that the function f is invariant over partitions of the input space X. That is, we can divide X into partitions $X = \bigcup X_a$ such that $\mathbf{x}, \mathbf{x}' \in X_a$ implies $f(\mathbf{x}) = f(\mathbf{x}')$. This concept of invariance could be used in learning-from-examples approach to object recognition to represent the fact that recognition is invariant to translations, scales, and rotations in the image plane. In this case, the partitions of X are collections of images of a particular object under different planar transforms. How do we modify the learning procedure so that the invariance hint can be "absorbed"? Keeping in the learning-from-examples framework, the invariance hint itself is represented using examples. The method for learning from examples of hints is discussed in the context of the standard backpropagation technique. For the case of a normal input-output pair $(\mathbf{x}, f(\mathbf{x}))$, backpropagation measures network output $y(\mathbf{x})$ and feeds the error $(y(\mathbf{x}) - f(\mathbf{x}))^2$ back through the network to modify weights. An example of the invariance hint is a pair $(\mathbf{x}, \mathbf{x}')$ from the same partition, and the error $(y(\mathbf{x}) - y(\mathbf{x}'))^2$ is fed back through the net. We can keep on generating examples of invariance hints as long as we can identify different examples from the same partition.

Mitchell and Thrun [95][128] have also developed a method for exploiting prior domain knowledge, a method called *explanation-based neural network learning* (EBNN). Prior domain knowledge is learned from previous experience with the problem, and is encoded as a neural network that is itself trained with input-output pairs. For example, in [128] the general notion of invariance under pose and lighting is learned for eventual use in training an object recognition system. Inputs to this network are pairs of images: pairs of the same object are positive examples while pairs of different objects are negative examples. To use this prior knowledge to help learn a new problem, such as the recognition of a particular object, the domain knowledge is used to help "explain" the input-output pairs $(\mathbf{x}, f(\mathbf{x}))$ for the new problem. By "explain", the domain knowledge can be used to estimate, given $\mathbf{x}$, both $f(\mathbf{x})$ and the partial derivatives of f with respect to the input $\mathbf{x}$. The key improvement to the learning procedure in EBNN lies in using the partials to speed up the learning, for learning a function is easier when both function values and its derivatives are available. For example, consider the invariance network with one of the two inputs fixed using the input $\mathbf{x}$ of a training example. This network is an approximation of the network that we wish to learn; we can differentiate to get the partials of f. Thus, the prior domain knowledge gives us additional information to learn f, namely the partial derivatives.

In the area of object recognition, both hints and EBNN discuss how prior knowledge of invariance can be used to assist in learning a particular object. Relative to our own work, hints is closer since the construction of additional examples is suggested, similar to our idea of creating virtual examples. The additional examples suggested by Abu-Mustafa are generated by applying a planar transform to a given example, basically changing translation, scale, and image-plane rotation. Such virtual examples, however, are not needed for our face recognition system since we *explicitly* normalize for translation, scale, and image-plane rotation by detecting facial features. Our virtual examples sample the remaining pose parameters, rotations "in depth". This can still be interpreted within the hints framework – only it is much more difficult to generate new examples since objects must be rotated in depth.

Since the object domain in this thesis is faces, the prior knowledge we use to generate virtual example views will be about the object class of faces. The resulting virtual views will be used as examples in a view-based, pose-invariant face recognizer, so let us now discuss the problem of face recognition in more detail.

1.3 Application: Face recognition

As already stated, the goal of this thesis is to explore the view-based approach to object recognition in a real application domain, face recognition. As a special case of object recognition, face recognition involves matching a new input face against a database of stored faces. That is, we start with a database or gallery of images of known faces. Given as input the image of an unknown face, which might be a digitized signal from a video camera or a digitized photograph, the problem is to identify the individual as someone in the database or to reject the input as unknown.

1.3.1 Recognition and related problems

The fact that face recognition specializes the recognition task to the class of faces turns it into a discrimination task. Usually the inputs to recognize have either been previously classified as faces or are known *a priori* to be faces, so the problem is to identify the individual from within the class of faces. This focuses the recognition task on the fine details needed to distinguish individuals from one another, on the differences between individual faces. In the psychology literature, identifying individual objects from a particular class is called recognition at the "subordinate" level.

The problem of *face detection* deals with the broader recognition task of distinguishing faces from non-faces. This is an example of the more abstract problem of categorization, where the goal is to assign general categories to objects, to label objects as faces, chairs, or cars, etc. In the psychology literature this is known as recognition on the "basic" level. On the computational side in computer vision, this is often cast as a detection task. The goal of object detection is to locate all instances of a particular class of object, faces in our case, even when the scene is cluttered with a variety of different object classes.

While a system that identifies faces in cluttered scenes should clearly employ a face detection algorithm to focus its attention on actual faces, in this thesis we address the identification problem. The identification problem is rich enough to allow us to explore the use of real and virtual views in view-based recognition. The need for face detection is avoided by taking face images against a uniform background and having the face take up most of the image (see Fig. 1-2). But even if these images were not available and we had to use cluttered scenes, there are existing face detection systems (Sung and Poggio [125], Sinha [122], Moghaddam and Pentland [96]) that could be used.

A related problem to face recognition is *face verification*: given an input face image and a proposed identity, verify that the face indeed belongs to the claimed person. This problem is less computationally demanding than full-blown face recognition since it does not require a sweep through the entire database of people. The focus is on rejection characteristics of the system, which involves reliably rejecting intruders while not falsely rejecting valid inputs.

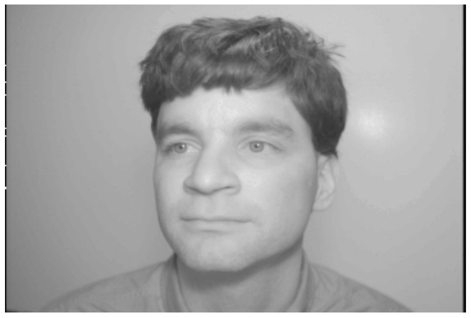


Figure 1-2: The problem of face detection is avoided by taking images of the face against a uniform background.

1.3.2 Motivation and difficulties

What makes face recognition an interesting problem? Consider the importance of the face in human culture. The human face is central to social interaction, as it is one of the cues we use to identify others, it displays our emotional state, and it is the center of attention during conversation. Since face recognition is an important skill used every day in normal social discourse, a natural question to ask is whether we can endow machines with the same ability. Face recognition is one of the many basic competencies that can be used to make computers behave intelligently.

To be more specific, consider the many applications of automatic face recognition. Virtually any application that currently requires badges, keys, or passwords for authenticating a person's identity could potentially use face recognition, even if only as a supplement to increase the security of current authentication measures. For example, in building security, a face recognizer could be used at the front entrance for automatic access control. They could be used to enhance the security of user authentication at ATMs, where cameras are commonly already in use. In the area of human-computer interaction, workstations with cameras would be able to recognize users, perhaps automatically loading the user's environment when he sits down in front of the machine. Finally, in law enforcement, face recognition could be used to match mug shots against databases of known criminals.

The importance of faces in the human visual system is underscored by findings in psychology and neurophysiology. Our proficiency with faces may be hardwired into our brains, as neurophysiological evidence for "face" cells has been uncovered in monkeys (Perrett, *et al.* [104], Desimone [48], Young and Yamane [143]). In a disorder called *prosopagnosia*, patients with certain types of brain damage lose their ability to recognize faces (Hanley, Young, and Pearson [65], Etcoff, Freeman, and Cave [52]).

What makes face recognition difficult? First, as with generic object recognition, the appearance of a particular face varies due to changes in pose, lighting, expression, and even factors like age and change in facial hair. Variation in pose and lighting conditions is a difficulty shared with the more standard problem of rigid object recognition, as faces are examples of 3D objects that change appearance when rotated in depth or lit differently. While pose and lighting changes are fairly well understood in the computer vision community, the nonrigidness of faces seen in expressions is only now being modeled, and factors like aging, make-up, and changes in facial hair are usually not even considered. Overall, the variability in appearance complicates the modeling of faces, for the recognizer either needs a face representation that is invariant to these factors, or it needs to model how these factors change facial appearance.

Stemming from the "subordinate" level nature of face recognition, a second difficulty is that faces form a class of fairly similar objects – all faces consist of the same facial features in roughly the same geometrical configuration. As a fine discrimination task, face recognition may require the use of subtle differences in facial appearance or the configuration of features.

1.3.3 Prior work

Prior work in face recognition has a history in the computer vision and pattern recognition communities going back over 20 years. While many face recognition systems have been proposed, most follow a typical sequence in terms of building the recognition system and recognizing new inputs. In designing the recognizer, the key decisions are choosing a representation for faces and an accompanying matching metric for comparing faces. For a given face image I, let $R(I)$ be its representation and $D(R(I_1), R(I_2))$ be the distance between images I_1 and I_2. To acquire a database of individuals known to the system, face images are taken of each person, converted to the face representation, and stored. Let us say that a total of n model images M_i are taken, where $1 \leq i \leq n$ and there may be more than one model image per person. To recognize a new input view I, the input face representation $R(I)$ is computed and matched against the model representations

$$i_{min} = \arg \min_{1 \leq i \leq n} D(R(I), R(M_i)).$$

The person identified by the recognizer is the person in model image $M_{i_{min}}$. In addition, some systems include the notion of rejecting the input if the best match is not good enough. This is commonly implemented by requiring the distance of the best match $D(R(I), R(M_{i_{min}}))$ to be below a certain threshold. In general, the rejection threshold in classifiers is either empirically determined or estimated by using techniques from statistical pattern recognition.

Ideally, one wants a face recognition system to handle as much variation as possible in terms of pose, lighting, and expression. However, in most prior work, especially the early systems, the model views M_i and inputs I were restricted to frontal pose, neutral expression, and fixed lighting conditions. This left some of the complexities of the problem unexplored, such as the problem's 3D nature from rotations in depth and nonrigidness from expressions. It has only been over the last few years that these restrictions have begun to be lifted by looking at multiple view-based systems and flexible matching procedures. Outside of the recognition problem, there have been recent studies on analyzing faces under different lightings (Hallinan [64]) and expressions (Essa [51], Yacoob and Davis [142], Beymer, Shashua, and Poggio [19]).

While the prototypical face recognition system deals with intensity images of frontal or near-frontal views, there are systems that are based on 3D range measurements and others that utilize the facial profile seen in side views of the face. More will be said about these systems in Chapter 2 on existing work.

In discussing the design and evaluation of existing face recognition systems, we focus on the following important issues. We only give an overview of the issues here; Chapter 2 provides a more thorough presentation and a listing of references.

1. *Input representation.* How are images of faces represented? Face representations, which to date have focused on viewer-based, 2D representations rather than 3D ones, fall into two classes, a geometrical features approach and a pictorial approach. In the geometrical features approach, first a set of facial features are detected, features such as the iris centers, nostrils, corners of the mouth, outline of the chin, etc. From the feature locations, geometric measurements are made and gathered into a feature vector. The resulting feature vectors have been fairly low-dimensional, usually 10- to 20-D, and the geometric features include measurements like point distances, angles, and curvatures.

   The second approach to face representation is pictorial in nature; the representation primitives are fairly "close" to the original face images. The template-based representation actually stores pixel intensity values from subimages, or *templates*, around the major facial features such as the eyes, nose, and mouth. The pixel values may be from the original intensity images or on versions of it preprocessed by gradient or Laplacian filters. A related filter-based approach applies filters such as the Gabor or Gaussian functions to a sparse set of image locations and represents images as a set of filter outputs. Another class of pictorial approach decomposes the grey level image as a linear combination of "eigenimages", which are derived from a principal component analysis of an ensemble of representative faces.

Comparing the geometric and pictorial approaches, recent momentum favors the latter. Implementing the pictorial approach is certainly simpler than the geometrical approach, which requires locating facial features. Recent systems for face recognition [129][20] [28][103] have chosen pictorial representations over geometrical ones. A comparative study by Brunelli and Poggio [28] favors the template-based approach over a typical feature geometry approach when recognition performance of both are compared on the same database. Indeed, the geometric approach may be too impoverished to sufficiently discriminate faces, especially as the database size gets large.

2. *Invariance to imaging conditions.* Is the recognizer designed to operate under changes in pose, lighting, and expression? Different approaches have been taken to handle the resulting variation in facial appearance.

   - *Intensity filtering.* Preprocessing the image using differential operators such as the gradient and Laplacian will introduce invariance to simple lighting changes. For example, changes that can be approximated by adding a constant to the image, such as changing the ambient illumination, will be factored out by differentiating. Using a normalized correlation metric, as we will describe later, accomplishes the same thing. Handling more complex changes, such as the lighting direction, requires more sophisticated methods (for example, see Hallinan [64]).
   - *2D geometrical invariants.* Transforming the image using the Fourier transform magnitude (Akamatsu, *et al.* [5]) or computing autocorrelation features (Kurita, *et al.* [82]) provides a representation that is invariant to 2D translation. The Fourier-Mellin transform, which in addition to 2D translation is also invariant to scale and image-plane rotation, has been explored by Fuchs and Haken [55][56].
   - *2D geometrical normalization.* If a couple of facial features can be located, such as the eyes, then the face image can be resampled using a similarity transform to normalize for the effects of translation, scale, and image-plane rotation. Since our face recognizer uses this method, more will be said about this in the next section.
   - *Multi-view representations.* As already mentioned with the view-based approach, some face recognition store many views per person to handle a range of rotations on the viewing sphere.
   - *Elastic matching.* von der Malsburg and collaborators [83] [91][141] use an elastic graph matching technique that is capable of matching model faces

with input faces even when they are separated by an out-of-plane rotation or difference in expression.

3. *Experimental issues.* While the previous two issues dealt with the design of the face recognizer, the following issues are central for its experimental evaluation.

   - *Recognition statistics.* Face recognizers are evaluated on a set of test images, images which are usually distinct from the example views stored for each person. These test images may contain people who are both in and out of the database, with views of the latter ideally being rejected by the recognizer as unknown. For the former group, the group of test images of people in the database, the *recognition rate* is the fraction of those images correctly identified by the system. Relatively high recognition rates, rates in the mid to upper 90%, have been reported on mid to large size databases (see Baron [11], Brunelli and Poggio [28], Cannon, *et al.* [35], Pentland, *et al.* [103]).

   - *Number of people in the face database.* The more people there are in the face database, the more difficult the discrimination task becomes. Intuitively one can think of our input space for representing faces as becoming more crowded with clusters of example views corresponding to each person. Most prior work in face recognition has dealt with databases on the order of tens of people. Recently, databases with hundreds and even thousands of people have become available. Examples include the new database being collected under the Army FERET program and a database collected by Pentland, *et al.* [103]. To be commercially viable for, say, security applications, it is generally agreed that face recognizers need to be proven on these larger databases.

   - *Variation in image test set.* As mentioned previously, face recognition is difficult because of the variability in appearance of a single face due to changes in pose, lighting, expression, and even changes in facial hair or the addition of paraphernalia such as hats or glasses. Most prior work has limited the scope of the problem by drawing both example and test views from a frontal pose, fixed lighting, and neutral expression. As we will describe in Chapter 2, more recent work is expanding the variation seen in test sets, thus demonstrating face recognition under more general imaging conditions.

Overall, the important experimental question currently being explored in face recognition research is whether the high recognition rates seen in earlier work

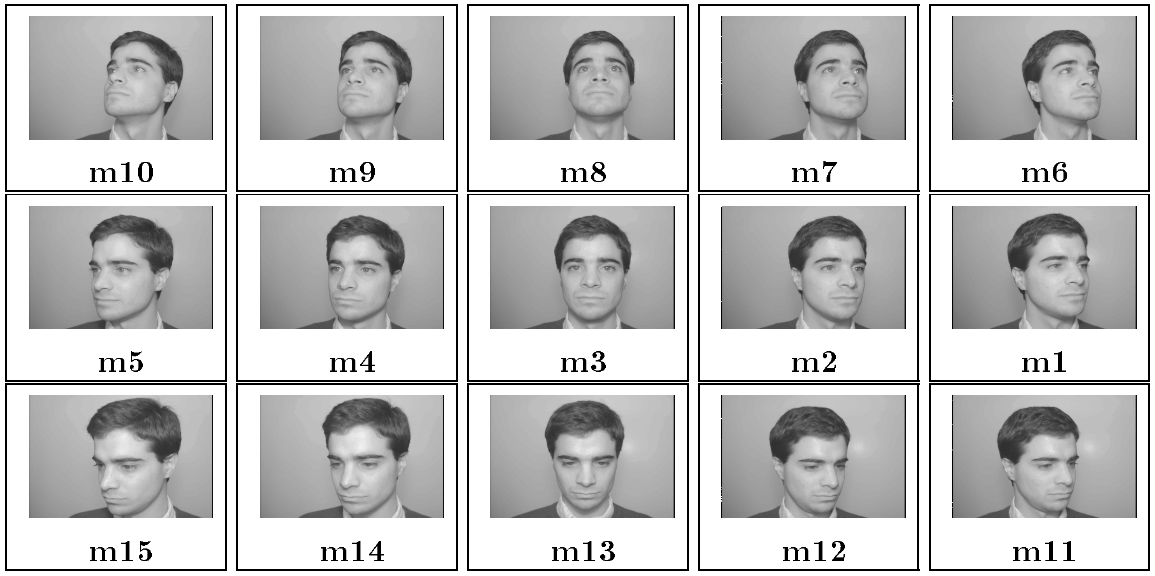


Figure 1-3: The view-based face recognizer stores 15 example views per person.

can be sustained when the databases are expanded and the variation in the test set is increased.

1.4 Our view-based, pose-invariant face recognizer

In this thesis we explore the problem of recognizing faces under varying pose. In other words, the input views presented to the recognizer for identification are not limited in pose to a frontal view, as has been the case for most prior work in face recognition. Input pose is allowed to fall within a range of acceptable poses, the difficult part of which is to handle rotations in depth. Our goal is to demonstrate that face recognizers can be extended to handle a range of rotations in depth. This can be seen as part of the longer term goal of building a face recognizer that works under a variety of poses, lighting conditions, expressions, etc.

1.4.1 View-based approach

Our pose-invariant face recognizer will use the view-based approach for recognition. Rotations in depth, or the rotations about the x- and y-axes in Fig. 1-1, will be handled by sampling a number of views on the viewing sphere. The recognizer will store 15 example views per person (Fig. 1-3), including 5 rotations about the y-axis and 3 rotations about the x-axis. Recall from our discussion of the view-based approach that a 2D matching algorithm is used to match input views against these stored example views. Our face recognizer will solve this 2D matching task by matching eyes and nose features between the input and example views. This will geometrically normalize the input for the effects of translation, scale, and image-plane rotation.

The range of acceptable poses for our recognizer is determined by the sampling of the viewing sphere in Fig. 1-3 and by the 2D geometrical normalization procedure. First, the sampling of the viewing sphere determines the range of rotation angles in depth, with rotations about the y-axis within the range $\pm 30°$ and rotations about the x-axis within the range $\pm 20°$. Keeping rotations about the y-axis within $\pm 30°$ ensures that both eyes will be visible, which is important since the eye locations will be used to geometrically normalize the input. Tolerance to the remaining pose parameters, translation, scale, and image-plane rotation, is determined by the feature finder used to locate the eyes and nose for geometrical normalization. Our feature finder, to be described in detail in Chapter 3, can handle image-plane rotations within $\pm 45°$ and any 2D translation. However, only minor variations in scale are currently allowed, a range of $\pm 20\%$ about an expected scale. The feature finder could be extended in a straightforward way, though, to extend its operable range of scales.

Besides the view-based approach, other approaches, namely the 3D model-based approach, could have been taken to the problem of pose-invariant face recognition. For example, textured 3D models of individual faces can be automatically constructed using specialized equipment such as the Cyberware scanner. These 3D models can be rendered under any desired pose using standard techniques from computer graphics. Compared to the view-based approach, however, the 3D approach is more complex since it requires

- specialized equipment for active depth sensing during the model acquisition phase, and
- a 3D rendering step to synthesize 2D views of the face from the 3D model.

The view-based approach, on the other hand, requires more memory to store the example views, so the question of the 3D model-based versus view-based approach can be framed as a trade off of complexity/computation versus memory. Overall, we are attracted to the view-based approach because of its simplicity, even if it is a relatively expensive one in terms of memory.

In this thesis we explore the view-based approach under two extremes in terms of the number of available example views for each person:

1. multiple example views available, and
2. one example view available.

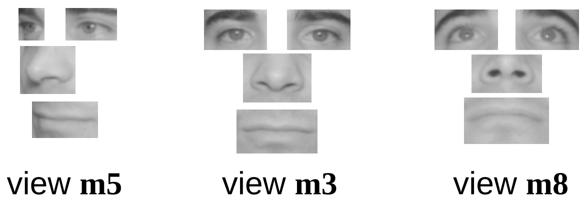


Figure 1-4: Faces are represented using templates of the eyes, nose, and mouth.

In the first case, we assume that the 15 views shown in Fig. 1-3 are available as examples for each person. This corresponds to the typical scenario for the view-based approach – lots of real data is available. But what if we only have one view per person, perhaps a digitized drivers license photograph or a passport photograph. Can we still use the view-based approach? In the second part of the thesis we assume that view **m4** is the single available view and we try to synthesize the remaining 14 views. These synthetic views, or *virtual views*, will be generating using prior knowledge of face rotation, knowledge that is represented in an example-based way from views of a prototype face. The earlier introduced methods of parallel deformation and linear classes will be the specific techniques we use to generate virtual views.

The two scenarios of the view-based approach, real views and virtual views, will both be examined using the same face recognition algorithm and the same image test set. We now give a quick overview of our specific face recognition algorithm.

1.4.2 Our face recognition system

The discussion of the details of our face recognition system is organized around the same basic issues we raised previously when introducing prior work in face recognition. These issues again are input representation, invariance to imaging conditions, and experimental issues.

Input representation

Motivated by the success of recent face recognition work that uses templates (Burt[32], Baron[11], Bichsel[20], Brunelli and Poggio[28]), we represent faces using templates of the major facial features, the eyes, nose, and mouth (see Fig. 1-4). These templates will be extracted and stored for each example view of each person in the database. The matching metric used to compare these templates to an input image is normalized correlation

$$r = \frac{<\mathbf{TI}> - <\mathbf{T}><\mathbf{I}>}{\sigma(\mathbf{T})\sigma(\mathbf{I})},$$

where $\mathbf{T}$ is the template, $\mathbf{I}$ is the subportion of image being matched against, $<>$ is the mean operator, and $\sigma()$ measures standard deviation. Normalized correlation is simply standard correlation $< \mathbf{TI} >$ normalized by subtracting off the means and dividing by standard deviation; the resulting coefficient r varies from -1 to $+1$. This normalization provides an invariance to grey level shifts of the template $\mathbf{T}$ of the form $a\mathbf{T} + b$, where a is a constant scaling factor and b is an additive constant. This kind of invariance may provide immunity to differences between template and image in the overall ambient lighting level or camera contrast.

Some of the remaining design issues for using templates are grey level preprocessing and overall template scale. For the issue of preprocessing, instead of using the intensity values from the original example images, the example images are preprocessed using filters such as the gradient, Laplacian, or Gabor functions before extracting the templates. Derivative-taking operators such as these assist with invariance to minor lighting changes as they subtract out additive constants. In addition, the Laplacian can help factor out the slowly varying low-pass characteristics of a local light source that illuminates some parts of the scene more brightly than others. The second issue, template scale, determines the resolution of the entire matching process, with smaller templates keeping less detail but enabling faster computation times. We shall evaluate the face recognizer for different combinations of template scales and image preprocessings.

Invariance to imaging conditions

How does our face recognizer handle variations in pose, lighting, and expression? The emphasis of our face recognition system is handling variation in pose. As previously mentioned, a view-based approach employing 15 example views per person is used to cover different views on the viewing sphere, or the rotations in depth. The remaining pose parameters, 2D translation, scale, and image-plane rotation, are factored out using a 2D geometrical registration procedure that is driven by the eyes and nose features.

To provide feature locations for the 2D registration procedure, a person- and pose-independent feature finder is first run to locate the two irises and one nose lobe feature. Fig. 1-5(a) shows an input view with the features detected by our feature finder. When using these features to register the input against an example view, say the example view shown in Fig. 1-5(b), the input is resampled using an affine transform that aligns the input eyes and nose features with those same features on

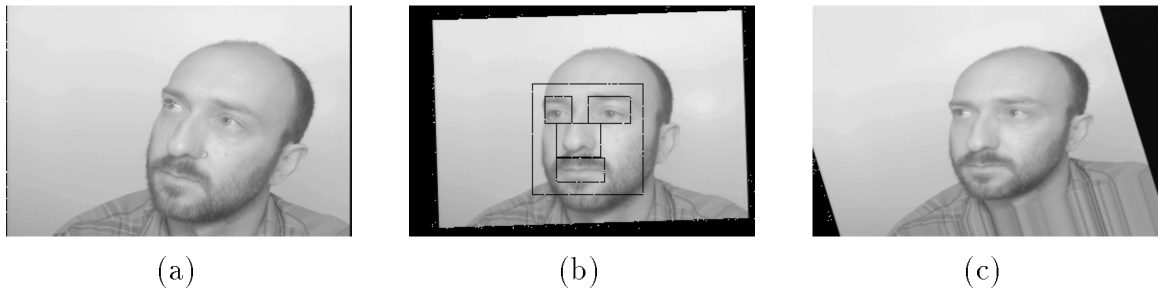


Figure 1-5: 2D registration step. Eyes and nose features in an input view (a) are brought into correspondence with an example model view (b), producing the new image (c) which is the original input resampled under an affine transform.

the example view (Fig. 1-5(c)). If the input and example views differ by only a 2D transform, the affine resampling will undo this transform. If the input and example views additionally differ by a rotation in depth, then the affine transform will use a combination of shears and nonuniform scales to distort the input view. This treats the input face as a textured planar object, where the plane is defined by the eyes and nose features.

After the feature finder locates the eyes and nose features, the face recognizer effectively searches over the rotation parameters in depth by trying to match the input against each example view of each person. The matching of the input to a particular example view consists of two main steps, the 2D registration procedure described above followed by a correlation step where example templates of the eyes, nose, and mouth are matched against the affine resampled input using normalized correlation.

While the emphasis of our face recognizer is pose-invariance, some parts of our design provide for invariance to minor variations in lighting and expression. As just mentioned for our template-based representation, some invariance to lighting conditions is achieved by preprocessing the example and input images and by using normalized correlation. But this is mostly for factors like overall illumination levels and contrast, and does not extend to handle changes in lighting direction. For minor variations in expression, our 2D registration procedure uses a secondary stage of processing after the affine transform to provide a finer registration based on optical flow. This secondary registration stage will be described in more detail in Chapter 4.

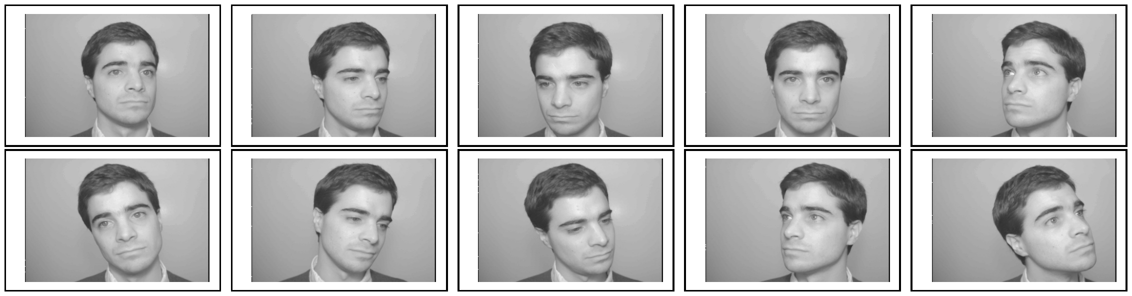


Figure 1-6: For each person, 10 test images are taken that sample random poses from the viewing sphere.

Experimental issues

To evaluate the use of real and virtual views in our face recognizer, we have collected an image data set of 10 test views per person. Shown in Fig. 1-6, the test views are taken under a variety of rotation angles both in and out of the image plane in order to test pose-invariant recognition. We currently have 62 people in the database, which is larger or comparable to most face databases in terms of number of people, although one recent database has thousands of people (Pentland, *et al.* [103]). To give a quick preview of recognition rates, when using real views it is 98%, and the best case scenario for virtual views is 85%.

1.5 Contributions

1.5.1 Main contributions

In the course of studying the problem of pose-invariant face recognition using real and virtual views, we hope to make the following main contributions:

- *View-based approach to pose-invariant face recognition.* Our face recognition system uses a strategy that combines geometrical normalization using a few facial features with template matching using templates from multiple views on the viewing sphere. Experimental results using real example views demonstrate the success of this approach when multiple views are available per person. This systematic study of pose-invariant face recognition helps to push forward the state of the art in face recognition, which prior to this work had mostly dealt with frontal or near-frontal views.

- *Virtual views.* In cases where only a limited number of example views per object are available, but prior knowledge is available about the class of object, then generating virtual views allows us to expand the example set. We demonstrate the application of two techniques, parallel deformation and linear classes, to the problem of generating virtual views of a face from just one example view. Using the combined set of one real and multiple virtual views, one obtains a higher recognition rate in our view-based face recognizer than using the single real view alone. In general, virtual views should increase the generalization performance of example-based learning techniques in the portions of the input space populated by the virtual examples.

1.5.2 Secondary contributions

The secondary contributions of the thesis are motivated by the feature correspondence requirements of the face recognizer and the process of generating virtual views:

- *Person- and pose-independent feature finder.* Our facial feature finder, which is run prior to the face recognizer, locates the two eyes and a nose lobe feature. As described in Chapter 3, it uses a large number of templates of the eyes and nose region to achieve person- and pose-independence. While our face recognition system uses the feature locations for geometrical registration of input and example views, the feature finder will be useful in many applications that process images of faces. For instance, the feature finder could be used to initialize a facial feature tracker, finding the feature locations in the first frame. This would be useful for applications like HCI and low-bandwidth teleconferencing.

- *Face "vectorizer".* In using images of prototype faces as prior knowledge for generating virtual views of a novel person, the most difficult requirement is to find interperson correspondence, defined as correspondence between the images of two different people. For example, parallel deformation requires mapping the face of the prototype onto the novel person. This mapping requires on the order of tens of feature correspondences rather than the three features located by the feature finder. Our face "vectorizer", to be described in Chapter 6, computes a dual representation of face shape and "texture", where shape refers to feature locations and texture to the modeling of intensity values. The shape component of the vectorizer will be used to find interperson correspondence.

1.6 Roadmap

The remainder of the thesis is organized as follows.

Chapter 2 on previous work gives a more detailed account of prior work in face recognition. To cover related work in generating virtual views, we also explore methods that use 3D models for face synthesis. Compared with using 2D images of a prototype, 3D models are an alternative way to express prior knowledge of faces.

The next two chapters, 3 and 4, discuss the face recognizer and the experimental results with real views. Chapter 3 focuses on the facial feature finder, which locates the two eyes and a nose feature. Chapter 4 describes the view-based face recognizer.

Chapter 5 introduces a "vectorized" image representation that will be used in generating virtual views. A novel, practical method for automatically computing the vectorized representation for faces will be discussed next in Chapter 6.

Chapter 7 on virtual views describes how we generated rotated virtual views of faces using the techniques of parallel deformation and linear classes. The results of using these virtual views in the view-based recognizer of Chapter 4 are then presented.

Chapter 8 closes the thesis with a discussion of future work and conclusions.

Appendix A discusses the face database we collected for the face recognizer, and Appendix B describes the mathematical details of the linear class approach for synthesizing virtual views.

Chapter 2

Previous Work

Within the engineering realm, the human face has been studied in both the computer vision and computer graphics communities over the last two decades. In computer vision and pattern recognition, images containing faces are processed to perform tasks like face detection and recognition. Computer graphics addresses the inverse problem, that of rendering realistic faces from modeling parameters of the face.

Both the analysis and synthesis tasks are relevant for this thesis. Since the main thesis topic is face recognition, the first part of this chapter explores face recognition in more detail than provided in the introduction. The second half of this thesis concentrates mostly on synthesizing virtual views of faces, so this chapter also discusses some prior work in the computer graphics synthesis of faces.

2.1 Face Recognition

In the introduction, we defined the problem of face recognition and gave an overview of the major issues facing existing face recognition systems. These issues were input representation, invariance to imaging conditions, and experimental issues. In this section, we further explore these issues, providing a more extensive list of references.

While our discussion of existing work will focus on recognizers that use intensity images of frontal views – the majority of face recognition work – there are some systems that use 3D range data or profile images, and some examples of this work will be listed at the end of this section.

2.1.1 Input representation

Comparing model and input faces boils down to performing distance measurements in the space used to represent faces. As current face recognition systems use fairly standard distance metrics like weighted norms and correlation, the main factor that distinguishes different approaches is input representation. There are two main approaches to input representation, a geometrical approach that uses the spatial configuration of facial features, and a more pictorial approach that uses an image-based representation.

There have been several feature geometry approaches, beginning with the seminal work of Kanade[73], and including Kaya and Kobayashi[76], Craw and Cameron[46], Wong, Law, and Tsang[139], Brunelli and Poggio[28], and Chen and Huang[38]. These feature-based systems begin by locating a set of facial features, including such features as the corners of the eyes and mouth, sides of the face and nose, nostrils, the contour along the chin, etc. These features are usually located using specialized heuristic procedures that are cued on edges, horizontal and vertical projections of the gradient and grey levels, and deformable templates (see Yuille, Hallinan and Cohen[144]). The spatial configuration of facial features is captured by a feature vector whose dimensions typically include measurements like distances, angles, and curvatures. For the systems listed above, the dimensionality of the feature vector varies from around 10 to 50. Craw and Cameron's system[46], which uses a novel feature vector, represents feature geometry by displacement vectors from an "average" arrangement of features, thus representing how a person differs from the norm. Once faces are represented by feature vectors, the similarity of faces is measured simply by the Euclidean distance or a weighted norm, where dimensions are usually weighted by some measure of variance. To identify an unknown face, geometry-based recognizers choose the model closest to the input image in feature space. So far, as we will discuss more in section 2.1.2, this approach to face recognition has been limited to frontal views, as the the geometrical measurements used in the feature vector are not invariant to face rotations outside of the image plane.

The second major type of input representation is pictorial in nature, representing faces by using filtered images of model faces. In template-based systems, the simplest pictorial representation, faces are represented either by images of the whole face or by subimages of the major facial features such as the eyes, nose, and mouth. Template images need not be taken from the original grey levels; some systems use the gradient magnitude or gradient vector field in order to get invariance to lighting. An input face is then recognized by comparing it to all of the model templates, typically using correlation as an image distance metric. Baron[11] uses normalized correlation on grey level templates. His system partially motivated the template-based approach of Brunelli and Poggio[28], which uses normalized correlation on the gradient magnitude. Gilbert and Yang[59] built a real time hardware implementation of the Brunelli-Poggio system that features a custom-built correlation chip. Burt[32] represents and matches templates using a hierarchical coarse-to-fine structure, and Bichsel's templates[20] uses the x and y components of the gradient.

Principal components analysis has been explored as a means for both recognizing and reconstructing face images. It can be read as an optimized pictorial approach, reducing the dimensionality of the input space from the number of pixels in the templates to the number of eigenpictures, or "eigenfaces" (Turk and Pentland[129]), used in the representation. To apply principal components, one must assume that the set of all face images is a linear subspace of all grey level images. The spanning set of eigenfaces, called "face space" by Turk and Pentland, is found by applying principal components to an ensemble of face images. Faces are represented by their projection onto face space. Turk and Pentland were first at applying principal components to face recognition. Akamatsu, *et al.*[5] first preprocesses the face image by the Fourier transform magnitude to get translation invariance. Craw and Cameron[46] applied principal components to "shape-free" faces, faces that have been warped to move feature points to standardized locations. Dalla Serra and Brunelli[117] used principal components on templates of the major facial features, achieving recognition rates comparable to correlation but at a fraction of the computational cost. Pentland, Moghaddam, and Starner [103] applied eigenfaces to a series of problems: recognition on frontal views in a large database of over 3000 people, recognition under varying left/right rotation, and detection of facial features using "eigentemplates". Kirby and Sirovich[77] have demonstrated that faces can be accurately reconstructed from their face space representation.

Besides principal components analysis, other analysis techniques have been applied to images of faces, generating a new, more compact representation than the original image space. Kurita, Otsu and Sato[82] represent faces by using autocorrelation on the original grey level images. 25 autocorrelation kernels of up to 2nd order are used, and the subsequent 25D representation is passed through a traditional Linear Discriminant Analysis classifier. Cheng, *et al.*[39] and Hong[67] have applied Singular Value Decomposition (SVD) to the face image where the rows and columns of the image are actually interpreted as a matrix. Cheng, *et al.* use SVD to define a basis set of images for each person, which is similar to "face space", but only that each person has their own space. Hong creates a low dimensional coding for faces by running the singular values from SVD through linear discriminant analysis. Ramsay, *et al.*[113] have used vector quantization to represent faces; after a face is broken down into its major facial features, the face is represented by a combination of indices of best matching templates from a codebook. The primary issue here is how to choose the codebook of feature templates. Nakamura, Mathur, and Minami [100] have used "isodensity maps" to represent faces. The original grey level histogram of the face is divided up into eight buckets, defining grey level thresholds for isodensity contours in the image. Faces are represented by a set of binary isodensity lines, and face matching is performed using correlation on these binary images.

Connectionist approaches to face recognition also use pictorial representations for faces (Kohonen[80], Fleming and Cottrell [53], Edelman, Reisfeld, and Yeshurun[49], Weng, Ahuja, and Huang[135], Fuchs and Haken[55] [56], Stonham[124], and Midorikawa[94]). Since the networks used in connectionist approaches are just classifiers, these approaches are similar to the ones described above. In multilayer networks of simple summating nodes, inputs such as grey level images are applied at an "input layer", and activity in the output layer, usually arranged as one node per object, determines the object reported by the network. By presenting a "training set" of model face images, the network is trained using a "learning" procedure that adjusts the network parameters. Among connectionist approaches to face recognition, the two most important issues are input representation at the input layer and the overall network architecture. As previously mentioned, the input representations are pixel-based, with [80], [53], [55], and [94] using the original grey level images. [135] uses directional edge maps, [124] uses a thresholded binary image, and [49] uses Gaussian units applied to the grey level image. A variety of network architectures have been used. A vanilla multilayer network trained by backprop, probably the most standard approach, has been explored by [53] and [94]. In a similar approach, [49] trains a radial basis function network using gradient descent. [80] and [55] use a recurrent autoassociative memory that "recalls" the pattern in memory closest to the applied input. [135] uses a multilayer "Cresceptron" that is patterned after Fukushima's Neocognitron[57]. In [124], which uses a binarized image as input, the network is a sum of a set of 4-tuple AND functions.

Hybrid representations that combine the geometrical and pictorial approaches have been explored. In Cannon, *et al.*[35], a 5D feature vector of feature distances and intensities is used as a "first cut" filter on the face database. The final matches are done by using a least squares fit of eye templates. In another hybrid approach, Lades, *et al.*[83] and Manjunath, Chellappa, and von der Malsburg[91] represent faces as elastic graphs of local textural features. Feature geometry is captured in the graph's edges, which store the distance between the two incident features. Pictorial information is represented at graph vertices by storing the results of Gabor filters applied to the image at feature locations. During recognition, the input face graph is first deformed to match the model graphs. Matches are evaluated by combining measures of the geometrical deformation and the similarity of the Gabor filter responses. While described in terms of flexible graphs, this approach can be read as representing and matching flexible templates.

Having explored the issue of input representation, let us now continue our discussion of existing face recognition systems by analyzing their invariance to imaging conditions and experimental issues such as recognition rates.

2.1.2 Invariance to imaging conditions

The wide variation in face appearance under changes in pose, lighting, and expression makes face recognition a difficult task. While existing systems do not allow much flexibility in pose, lighting, and expression, most systems do provide some flexibility by using invariant representations or performing an explicit geometrical normalization step.

Invariant representations, representations that do not change when the input parameters change, can handle variations in lighting to a limited degree. For instance, by filtering the face image with a bandpass filter like the Laplacian, one can achieve some invariance to lighting conditions. Assuming that the image content due to lighting is lowpass, bandpass filtering should remove the lighting effects while still preserving the higher frequency texture information in the face. The assumption that lighting effects are lowpass breaks down, however, when there are cast shadows on the face, which usually happens when the face is illuminated from the periphery.

To provide shift invariance, some systems preprocess images using the Fourier transform magnitude or autocorrelation. This only handles the translational pose parameters, requiring other mechanisms to handle the rotations and scale parameters. Using a standard approach used in optical processing systems for invariance (e.g. ATR), Fuchs and Haken[55][56] factor out the image-plane rotation and scale parameters in addition to the translational parameters. First, they take the Fourier transform magnitude, which provides shift invariance. Next, the Cartesian image representation is transformed into a complex logarithmic map, a new representation where scale and image-plane rotation in the original image become translational parameters in the new space. Taking the Fourier transform magnitude again then provides invariance to scale and image-plane rotation. Using invariant representations to handle the remaining pose parameters, rotations out of the image plane, on complex 3D textured objects like faces has not yet been tried.

By finding at least two facial features – usually the eyes in existing systems – the face can be normalized for translation, scale, and image-plane rotation. In feature geometry approaches, distances in the feature vector are normalized for scale by dividing by a given distance such as the interocular distance or the length of the nose. In template-based systems, faces are often geometrically normalized by rotating and scaling the input image to place the eyes at fixed locations. These approaches, of course, cannot handle rotations outside of the image plane. The normalization step reduces pose space from its original 6D formulation to a 2D space of rotations out of the image plane. In a recognizer that allows general pose, rotations on the viewing sphere still need to be handled.

Most face recognition systems are not designed to handle changes in facial expression or rotations out of the image plane. By tackling changes in pose and lighting with the invariant representations and normalization techniques described above, current systems treat face recognition mostly as a rigid, 2D problem. There are exceptions, however, as some systems have employed multiple views and flexible matching strategies to deal with some degree of expression and out-of-plane rotations. In Akamatsu[5], four slightly rotated model views (up, down, left, right) are used in addition to a frontal view. In Otsu[82], up to 116 people with 50 images/person are used in building a Linear Discriminant Analysis classifier. These training images are extracted from a videotaped session with each person and cover different minor head rotations. Flexible graph matching techniques have also been used ([91], [83]) to enable matching one frontal model view to rotated views and views with facial expressions. What distinguishes my approach from these techniques will be a wider allowed variation in viewpoint and the use of prototype face knowledge to generate "virtual" model images.

2.1.3 Experimental issues

The evaluation of face recognition systems is largely empirical, requiring experimental study on a set of test images. As we mentioned in the introduction, the important issues in the experimental evaluation of face recognition systems are the recognition rate, the number of people in the database, and the variation in the test views. In this section we explore these issues in more detail.

Experimental studies in face recognition usually start by collecting a set of face image data with more than one view per person and dividing it into modeling and testing sets. The modeling data is processed to extract the system's representation for faces, whether it be a geometry feature vector or a set of templates. In connectionist approaches, the modeling images are used to train a network-based classifier.

After models of the faces are so constructed, recognition statistics are compiled by running the system on an image test set. There are two levels of testing, depending on whether the system includes the notion of rejecting inputs that are a poor match to the database.

1. *No rejection ability.* In the simpler case where rejection is not included, the system is tested only on images of people in the database. The potential for error here is a *substitution* error, mistaking one database person for another. The relevant recognition statistics are the recognition rate, which is the fraction

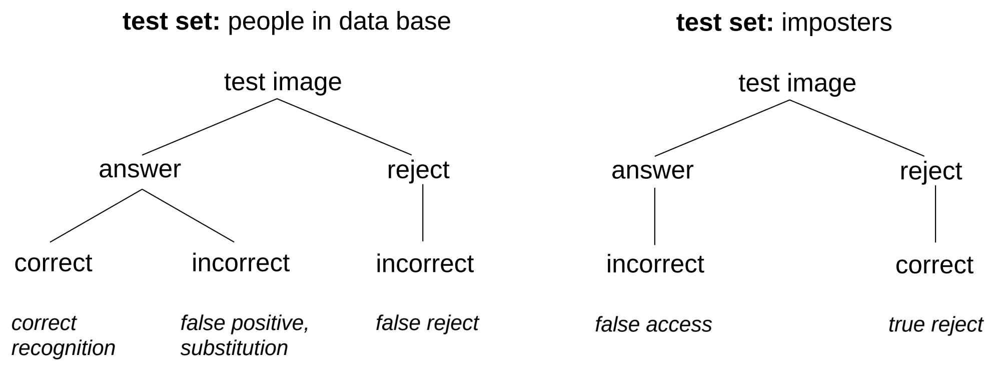


Figure 2-1: If the face recognizer includes the notion of rejecting inputs that are poor matches, it is evaluated on two test sets of images to collect recognition statistics: a set of people inside the database and a set of imposters. Different outcomes for these two test sets are listed above.

of inputs correctly recognized, and the substitution rate, the fraction of inputs falsely identified by the system.

2. *Poor matches rejected.* If a rejection capability is included, then testing is usually expanded to include imposters, or people from outside the database. As shown in Fig. 2-1, recognition statistics are collected over two separate groups of testing data, a group of database people and a group of imposters. For the test group of database people, we add a new statistic called the false rejection rate, which is the fraction of inputs that are falsely rejected by the system. Thus, the system can now err by either making a substitution or a false reject. For the second testing group of people outside the database, the relevant recognition statistic is the false access rate, or the fraction of images that are not rejected and hence mistakenly recognized as faces from the database.

One can trade off the rejection and recognition statistics by varying how strict the rejection criterion is. Using a stricter rejection criterion will increase the number of rejections, as inputs now need to be a closer match to the database to be accepted. Consequently, the false rejection rate will increase; the recognition rate, substitution rate, and false access rates are expected to go down. Notice that as one makes the rejection criterion more strict, the false access rate decreases, which is good. However, the recognition rate also decreases, which makes the system less effective. How to choose the proper rejection level for this trade off depends on the application. For an application like automated building access, one wants to minimize false accesses. Further, one doesn't particularly mind the increased inconvenience of a lower recognition rate (a user that is repeatedly rejected could call a guard for manual verification), so a strict rejection criterion should be used. Similarly, in systems that can grab several images of the person to identify, the rejection rate is not so important because the system has several attempts at recognition. On the other hand, for a lower security application such as human-computer interaction, one is less concerned with false accesses. Further, the user does not wish to be annoyed by false rejections, so a more liberal rejection criterion should be used.

Some face recognition systems have achieved good recognition rates. The early template-based system of Baron[11] reached an impressive 100% recognition rate on a database of 42 people. To test system rejection ability, the recognizer was tested on 108 faces from outside the database, with a resulting false access rate of 0%. Brunelli and Poggio's template-based system[28] achieved a recognition rate of 100% on frontal views of 47 people. The system of Cannon, *et al.*[35] was tested on a database of 50 people and reached a recognition rate of 96%. Turk and Pentland[129] report a 96% recognition rate when their system, which uses a database of only 16 people, is tested under varying lighting conditions. Akamatsu[5] reports 100% recognition, but on a smaller size database of only 11 people. Pentland, Moghaddam, and Starner[103] report a recognition rate of 95% on a database of over 3,000 people. Otsu[82] and Bichsel[20] give plots of recognition rate versus false access rate, but the results are not that impressive: to reach a recognition rate in the upper 90%'s, the false access rate also climbs to unacceptable levels.

Needless to say, these recognition statistics are meaningful only if the database of model faces is sufficiently large. Face recognizers that do well on small databases do not necessarily scale up to larger databases. This is one area where many studies in face recognition have been lacking; often, small databases of less than ten model faces have been used. While there is no consensus on the sufficient size of the model database, some of the more recent approaches ([82], [20], [91]) have used databases on the order of 70 people or more. Pentland, Moghaddam, and Starner[103] have the largest database in the research community – 7,562 images of over 3,000 people. A face database is being collected under the Army FERET program on face recognition. As of August, 1994, the database had over 500 people with up to several views per person. If we take a cue from the potential applications of face recognition when considering database size, databases on the order of 100's are certainly useful (e.g. automated building access), but other applications (e.g. law enforcement) would probably need larger databases on the order of tens of thousands.

Another important experimental issue is the variability of the test set. Since there will always be some amount of variation in pose and expression in a real face recognition system, experiments should ideally use a variety of test images per person sampling small changes in pose and expression. Existing face recognition work has varied in this regard. [82] uses 50 testing views per person extracted from a videotaped session where the subject was asked to rotate his head up, down, left, or right. The database of von der Malsburg and collaborators [91][83] includes frontal views, views rotated 15° to the right and views with varying expression. Pentland, Moghaddam, and Starner's[103] database is fairly controlled for pose and lighting, but expression and paraphernalia is allowed to vary. On the other hand, in [67], only one photograph is used for both modeling *and* testing, with scans of the photo at different offsets providing different images. In my recognition experiments, I take 10 test shots per person covering different rotation angles on the viewing sphere.

2.1.4 Related work

While our discussion of existing work has focussed on the domain of frontal views and intensity images, facial analysis and recognition has been tried with profile views of the face (Kaufman and Breeding[75], Harmon, *et al.*[66], Wu and Huang[140], Campos, Linney, and Moss[34]) and with other modalities, such as 3D depth data (Lapreste, Cartoux, and Richetin[85], Lee and Milios[86], Nagamine, Uemura, and Masuda[99], Gordon[61]).

In the profile view work, the face is imaged against a uniform background, making detection of the profile simple. Next, the profile is represented by a low dimensional (10-20D) vector of features extracted from the profile. In [75], vector components are autocorrelation coefficients on the binarized silhouette image. [66] uses a vector of distances, angles, areas, and curvatures on a set of automatically located fiducial points along the profile. In [140], after using a B-spline to segment the profile into 5 segments, the vector components measure segment lengths, angles, curvatures, and symmetry. Finally, while not working on recognition, [34] uses inflection points from a scale space analysis to segment the profile.

In the 3D range work, a depth map of the face is acquired by scanning the face with an active depth sensor. Face surface curvature is a key feature utilized by these methods. [85] coverts the range data into a 9D feature vector, the components of which measure distances between curvature features along the profile. In [86], the extended Gaussian image is used for representing and matching convex feature regions of the face. [99] represents faces by extracting portions of the range data along curves of intersection with the 3D data. Horizontal and vertical lines, as well as circles, are used as the intersection curves. Finally, [61] first finds high level curvature features such as the nose ridge and eye corner cavities and then creates a feature vector of distance and curvature measures between the high level features.

2.2 Synthesizing faces

To motivate our discussion of synthesizing faces, let us review the introductory comments about the second half of the thesis. The second half of this thesis addresses the problem of pose-invariant face recognition when only one view of each person's face is available. This problem is reduced to the multiple view scenario by synthesizing virtual views of each person. That is, the augmented set of one real and multiple virtual views will be used as example views in a view-based approach to the problem. Virtual views are synthesized using prior knowledge of facial rotation in depth. In what ways can one represent this kind of prior knowledge? While we use example images of a prototypical faces undergoing rotations in depth, a more traditional approach would have been to use a generic 3D model of the face. Since this method competes with our example-based method, in this section we briefly review 3D synthesis techniques for faces.

3D models combined with texture mapping from real faces have been used to synthesize images of faces under varying pose and expression. This has been explored in the computer graphics and computer vision communities and by researchers in low bandwidth teleconferencing (Essa and Pentland[51], Aitchison and Craw[3], Kang, Chen, and Hsu[74], Akimoto, Suennaga, and Wallace[6], Anderson and Dippe[7], Oka, *et al.*[101], Waters and Terzopoulos[127][134], Aizawa, Harashima, and Saito[4], Choi, Harashima, and Takebe[40], and Williams[136]). In these 3D modeling techniques, the shape of the face is represented either by a polygonal model or by a more complicated multilayer mesh that simulates tissue. After an example view of the face is texture-mapped onto the 3D model, new views of the face under changes in pose can be generated by rotating the 3D model and reprojecting to a 2D image. Faces are texture mapped onto the 3D model in one of two ways, either by specifying corresponding facial features in both the image and 3D model or by recording both 3D depth and color image data simultaneously by using specialized equipment such as the digitizer from Cyberware.

While some techniques use the 3D model to only generate different views of the face ([3], [74], [6]), others add mechanisms to alter facial expression as well. One method for altering facial expression uses interpolation between different views of the face under different expressions ([7], [101]). The other common approach for changing expression deforms the 3D model. [127][134], who model the face in 3D with a multilayer tissue model, translate expressions into muscle movements, which are simulated in the tissue and deform the upper layers of the skin. The teleconferencing systems of [4] and [40] and the animation system of [136] deform the 3D model simply by moving vertices in ways that mimic or track facial muscles.

As these expression-generating approaches are designed with teleconferencing or animation in mind, most of them track the expressions of a performer and then synthesize the same (or different) face with the same expressions. The tracking information is either low-level information such as the locations of major facial features or the "action units" of FACS (Facial Action Coding System) developed by Ekman and Friesen[50]. FACS consists of "action unit" parameters like "cheek raiser" or "lip corner puller". The approaches of [7] and [134] are not performance-driven, generating different expressions by hand tweaking the 3D model.

2.3 This thesis and prior work

How should one place this thesis with regard to prior work in face recognition and face synthesis? First, with regard to face recognition, our pose-invariant face recognizer pushes the state of the art forward in terms of handling rotations both in and out of the image plane. For each person in the database, rotations in depth are handled using a view-based approach that samples a set of 15 views on the viewing sphere. The representation for each example view in the database is template-based, thus building on the success of prior template-based face recognizers for frontal views [11][28].

Second, for synthesizing virtual views, our technique uses 2D views of prototypical faces rather than 3D models to express prior knowledge of facial rotations in depth. The motivation for staying in 2D rather than working in 3D is the potential for exploiting a shortcut that avoids the complexities of 3D modeling such as acquiring 3D models and texture mapping. One 2D approach we explore for virtual view synthesis uses simple 2D warping operators.

Part I

Face Recognition Using Real Views

Chapter 3

Feature detection and pose estimation

The first stage of processing in the proposed face recognition architecture is a person-independent feature finding and pose estimation module. As mentioned in the introduction, the kind of facial features sought by the feature finder are the two eyes and at least one nose feature. The locations of these features are used to bring input faces into rough geometrical alignment with the database example views. Pose estimation is used as a filter on the database views, selecting only those views whose pose is similar to the input's pose. By pose estimation we really mean an estimate of the rotation angles out of the image plane since feature locations have already been used to normalize for position, scale, and image-plane rotation. Pose estimation is really an optimization step, for even in the absence of a robust pose estimator, the system could still test the input against all example views of all people.

Before describing the details of our feature finder, let us first review some of the existing work in detecting facial features.

3.1 Previous work

Facial feature detection, for the most part, is the problem of locating the major facial features such as the eyes, nose, mouth, and face outline. Some researchers have also addressed the issue of characterizing facial features, usually with the parameters of a model fit to the feature. While most feature detection efforts are motivated by the need to geometrically normalize a face image prior to recognition, other applications of facial features include face tracking and detecting faces in cluttered images.

Most research to date has taken one of three major approaches, a parameterized model approach, a pictorial approach, and the use of grey level interest operators. In one parameterized model approach, deformable template models of individual facial features are fit to the image by minimizing an energy functional (Yuille, Hallinan, and

Cohen[144], Hallinan[63], Shackleton and Welsh[118], Huang and Chen[69]). These deformable models are hand constructed from parameterized curves that outline subfeatures such as the iris or a lip. An energy functional is defined that attracts portions of the model to preprocessed versions of the image – peaks, valleys, edges – and model fitting is performed by minimizing this functional. A related model-based approach fits a global head model constructed from tens of feature locations (Bennett and Craw[15], Craw, Tock, and Bennett[47], Cootes, *et al.*[44]) to the image by varying individual feature locations. Terzopoulos and Waters [127] have used the active contour model of snakes to track facial features in image sequences.

In the pictorial approach, a pixel-based representation of facial features is matched against the image. This representation may be templates of the major facial features (Bichsel[20], Baron[11], Burt[32], Poggio and Brunelli[28]), an "eigentemplate" decomposition following the eigenface recognition approach (Pentland, *et al.* [103]), or the weights of hidden layer nodes in neural networks (Vincent, Waite and Myers[133]). For the template-based systems, correlation on preprocessed versions of the image is the typical matching metric. The eigentemplate approach uses a "distance from feature space" metric, which measures the distance between a subimage being analyzed and its projection onto the eigentemplate space. The neural network approaches construct a network where implicit feature templates are "learned" from positive and negative examples.

Another major approach to facial feature finding is the use of low level intensity-based interest operators. As opposed to the model-based and template-based approaches, this approach does not find features with semantic content as, say, an eye, nose or mouth detector does. Instead, the features are defined by the local grey level structure of the image, such as corners (Azarbayejani, *et al.* [8]), symmetry (Reisfeld and Yeshurun[115]), or the "end-inhibition" features of Manjunath, Shekhar, Chellappa, and von der Malsburg[92], which are extracted from a wavelet decomposition of the image.

3.2 Overview of our method

While techniques already exist for finding facial features, no existing system has been demonstrated for our desired range of rotation angles, which includes rotations both in and out of the image plane. Thus, we need to build a system that addresses this issue. As just mentioned, existing methods for finding facial features with semantic content (i.e. the eyes or nose, as opposed to, say, a grey level interest operator) tend to fall into one of two categories, a pictorial approach and a model-based approach. In the model-based approach, however, the models and fitting procedures are usually ad hoc and require experimentation to fine-tune the models. The amount of work is manageable for one view but might become tedious as models and fitting rules for different views on the viewing sphere are developed. Thus, we chose to explore a template-based approach for our feature finder, primarily for its simplicity.

To serve as the front end of a pose independent face recognizer, the feature finder must, of course, handle varying pose and be person independent. The current system addresses these requirements by using a large number of templates taken from multiple poses and from different people. To handle rotations out of the image plane, templates from different views on the viewing sphere are used. Templates from different scales and image-plane rotations can be generated by using standard 2D rotation and scaling operations. To make the feature finder person independent, the templates must cover identity-related variability in feature appearance (e.g. tip of nose slanted up versus down, feature types specific to certain races). I use templates from a variety of exemplar faces that sample these basic feature appearances. The choice of exemplars was guided by a simple clustering algorithm that measures face similarity though correlation.

Our feature finder, then, entails correlation with a large number of templates sampling different poses and exemplars. To keep this search under control, we use a hierarchical coarse-to-fine strategy on a 5 level pyramid representation of the image. In what follows, level 0 refers to the original image resolution while level 4 refers to the coarsest level. The search begins by generating face location hypotheses at level 4, where the pose parameters are very coarsely sampled and only one exemplar is used. Exploring a level 4 hypothesis is organized as a tree search through the finer pyramid levels. As processing proceeds to finer levels, the pose parameters are sampled at a higher resolution and the different exemplars are used. A branch at any level in the search tree is pruned if the template correlation values are not above a level-dependent threshold.

The tree searching strategy starts out as a breadth first search at the coarser levels where the correlation scores are not entirely reliable. As processing reaches lower levels in the pyramid, correlation scores become more reliable and the search strategy switches to depth first. Search at levels 4 and 3 is breadth first: all possible level 3 hypotheses are generated from all level 4 hypotheses and then sorted by correlation score. Then the search strategy switches to a depth first search of level 3 hypotheses. If any leaves in the search tree (at level 0) pass the template correlation threshold tests, then the search is terminated – no more level 3 hypotheses are explored – and the leaf with the highest correlation scores is reported. Fig. 3-1 depicts the hierarchical search process, where crossed out hypotheses have failed the correlation threshold tests and the final answer is circled.

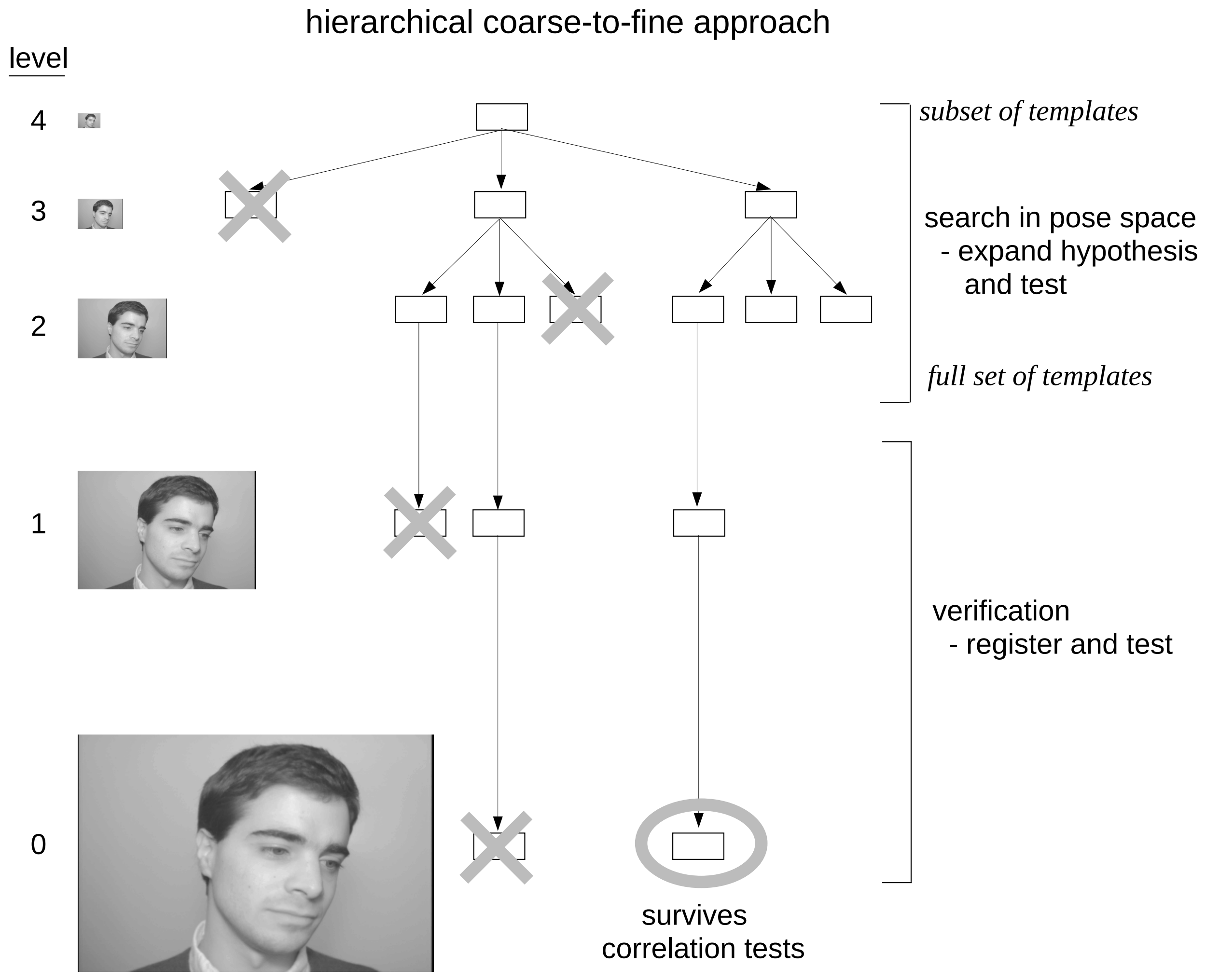


Figure 3-1: Hierarchical processing in our template-based feature finder. Search at the two top levels is performed breadth first, with all level 4 and 3 hypotheses first being generated and then sorted. The sorted level 3 hypotheses are then expanded depth first. The first level 0 hypothesis to survive the correlation tests (circled) is returned by the system. The figure shows only one of the many level 4 hypotheses, so the actual picture should have many of the above trees, one for each level 4 hypothesis.

3.3 Hierarchical processing

Search over different poses and exemplars through the 5 levels of the pyramid is organized as follows. At the coarsest level, level 4, the system is trying to get an estimate of the overall position of the face, so a bank of 30 different whole-face templates are correlated over the entire image. Because the resolution at this pyramid level is very coarse – the interocular distance is only around 4 pixels – the pose parameters can be sampled very coarsely, and only one exemplar is used. Currently, the system uses 5 left/right rotations (-30, -15, 0 15, 30), three image-plane rotations (-30, 0, 30), and two scales (interocular distances of 3 and 3.75). Local maxima above a certain threshold in the correlation scores generate face location hypotheses, which are explored by refining the search over pose parameters at the mid levels resolutions, levels 3 and 2.

When a pose hypothesis is being refined at level 3 or 2, pose space is explored at a higher resolution in a small neighborhood around the coarser pose estimate of the previous level. At level 3, for instance, the 5 left/right viewing sphere angles are expanded to include 3 up/down rotations (-20, 0, 20), bringing up to 15 the number of viewing sphere angles explored. Also at level 3 the image-plane rotation parameter is sampled at twice the resolution of level 4, now including 7 different rotations at 15 degree increments. The different exemplars are also tested. As mentioned before, pose space is explored in a small neighborhood around the coarse estimate of the previous level, so a level 4 hypothesis is examined at level 3 by searching over 3 up/down rotations, 3 image-plane rotations, and the different exemplars (currently 6) in a neighborhood around the level 4 correlation maxima. Pose hypotheses from levels 3 through 0 keep track of how all exemplars match the image at that pose.

For each of these level 3 hypotheses, search at level 2 occurs only if the template correlation is above a certain threshold. At level 2, the resolution of image-plane rotations is doubled again to every 7.5 degrees (for a total of 15 rotations from -52.5 to 52.5) and the search over the 3 up/down rotations is repeated. For level 2 hypotheses surviving the threshold test on the correlation values, the resolution of the image is high enough to allow estimating the locations of features, in this case the two irises and a nose lobe.

The repetition of the up/down rotation search on level 2 is done to increase the flexibility of the search – it is not always possible to make a choice on the up/down rotation at level 3, but including the extra up/down rotation templates at that level helps to assure that true positives are not rejected by the thresholding step. In general, the level for which the decision for a pose parameter is made may either be hard to estimate or person-dependent, so while repeating a search at two adjacent

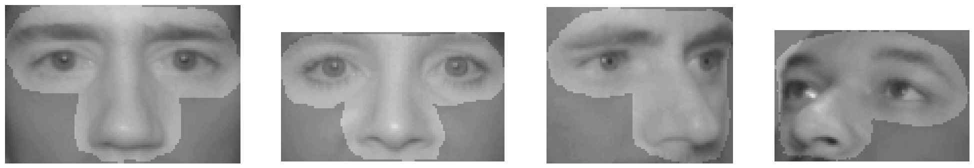


Figure 3-2: Example templates of the eyes and nose used by the feature finder.

levels may increase running time, it also increases system flexibility.

Processing at the finest levels of the pyramid, levels 1 and 0, are essentially verification steps. Level 2 hypotheses provide relatively good estimates of feature locations, and the finer levels use the eye locations to geometrically align the templates and image before correlating with templates. No further search over pose space is performed. The correlation tests at these levels serve to weed out any remaining false positives; hypotheses surviving level 0, which is at the resolution of the original image, are assumed to be correct and cause termination of the depth first search.

3.4 Template matching

Templates are manually chosen from 15 modeling images of the exemplars covering the viewing sphere. A special mask-defining program is utilized to draw template boundaries over the example modeling images. As templates are defined by these binary masks, templates can be tailored to tightly encircle certain features, not being limited to square regions. Actual templates used by the feature finder vary according to the level of processing. At level 4, the system is trying to get a general estimate of the face position, so full face templates are used, templates that run from above the eyebrows to below the chin. At finer resolutions the feature finder uses multiple templates that cover smaller areas; see Fig. 3-2 for some example templates. At level 3, to handle bangs vs. no bangs in the input, we use two types of templates with slightly different coverage above the eyes region. The second template from the left in Fig. 3-2 handles cases where where bangs come down to the eyebrows and obscure the skin above the eyebrows. The template on the far left handles cases where the bangs do not come down to the eyebrows. At level 2, the same eye/nose masks at level 3 are used, but the template is broken up into two eye and one nose subtemplates. At level 1, the same eye/nose masks are again used, but each eye and the nose are themselves vertically divided into two subtemplates, which yields 6 subtemplates total. Level 0 uses the subtemplate set of level 1 augmented by a circular subtemplate centered around the iris center or nose lobe feature.

The correlation thresholding test is based on eye and nose features, their subtemplates, and the fact that a pose hypothesis keeps track of the different exemplars. For a particular exemplar eye or nose feature, the correlation thresholding test requires that all subtemplates of the eyes and nose features exceed the threshold. For a pose hypothesis to pass the thresholding test, there must be some combination of passing eye and nose templates; the passing templates need not come from the same exemplar. This mixing of eye and nose templates across exemplars increases the flexibility of the system, as a face whose eyes match only exemplar A and whose nose matches only exemplar B will still be allowed.

Template matching is performed by using normalized correlation on processed versions of the image and templates. Normalized correlation follows the form

$$r = \frac{< \mathbf{TI} > - < \mathbf{T} >< \mathbf{I} >}{\sigma(\mathbf{T})\sigma(\mathbf{I})}$$

where $\mathbf{T}$ is the template, $\mathbf{I}$ is the subportion of image being matched against, $< \mathbf{TI} >$ is the normal correlation of $\mathbf{T}$ and $\mathbf{I}$, $<>$ is the mean operator, and $\sigma()$ measures standard deviation. We hope that normalized correlation will give the system some invariance to lighting conditions and the dynamic range of the camera, as the image mean and standard deviation are factored out. Correlation is normally carried out on preprocessed versions of the image and templates, again to provide for some invariance to lighting. While we have explored the x and y components of the gradient, the Laplacian, and the original grey levels, no preprocessing type has stood out as the best. Performing correlation using these different preprocessings and then summing the result, however, empirically yields more robust performance than any single type of preprocessing. Thus, the current system performs separate correlations using the grey levels, x and y components of the gradient, and Laplacian, and then sums the results.

At higher resolutions in the pyramid, the details of individual features emerge. This might foil the matching process because the features in the input will not precisely match the templates due to differences in identity and pose. For instance, the features in the input may not sufficiently close to *any* of the exemplar features, or the input features may be from a novel pose that is in between the template modeling views. In order to bring the input features into a better correspondence with the templates, we apply an image warping algorithm based on optical flow to "warp" the input features to make them look like the templates. First, optical flow is measured between the input features and the template using the hierarchical gradient-based scheme of Bergen and Hingorani[18]. This finds a flow field between the input feature and template, which can be interpreted as a dense set of correspondences. The input

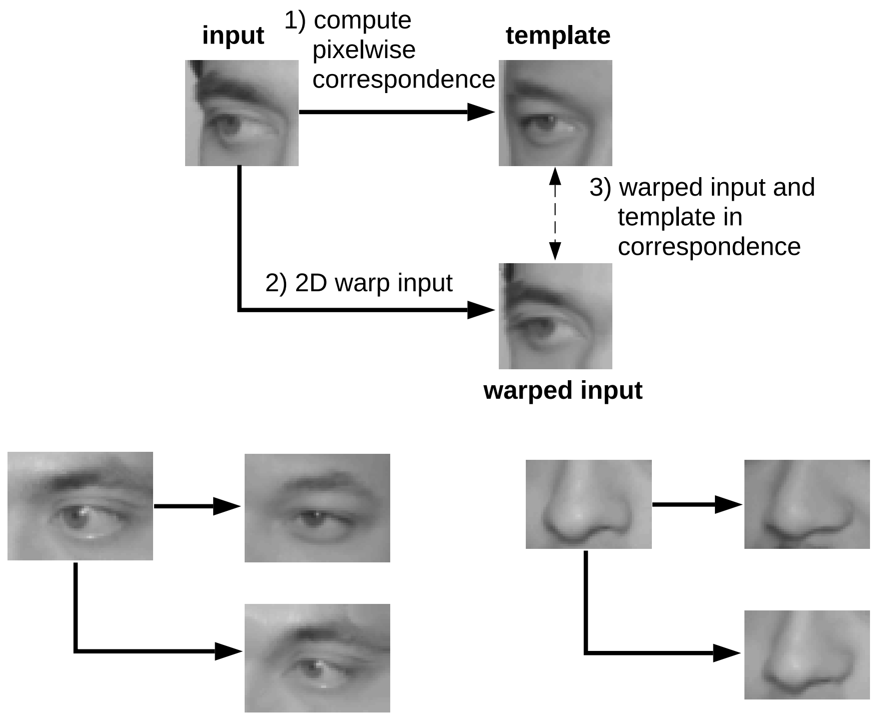

input
1) compute pixelwise correspondence
template
3) warped input and template in correspondence
2) 2D warp input
warped input

Figure 3-3: In the feature finding process, an extracted portion of the input (1) is brought into pixel level correspondence with a template using an optical flow algorithm. The input is then warped (2) to make it mimic the geometry of the template (3).

feature, as shown in Fig. 3-3, is then geometrically warped using the flow field to make the input feature mimic the shape of the template. This helps to compensate for small rotational and identity-related differences between the input features and templates. Correlation is performed after the image warping step.

Final feature locations are determined from a successful level 0 match returned by the depth first search. Feature points at the center of the irises and the nose lobes, which are manually located in the templates, are mapped to the corresponding points in the input image using the correspondences from optical flow. Fig. 3-4 shows the features located in some example test images. It is interesting to note that because correspondence from optical flow is dense, we could actually detect more than three feature points once we have brought our eye and nose templates into correspondence with the image; all we have to do is manually specify more points in the exemplar templates. We stop at three points because that is all that is needed to specify the affine transform used by the geometrical alignment stage in the recognizer.

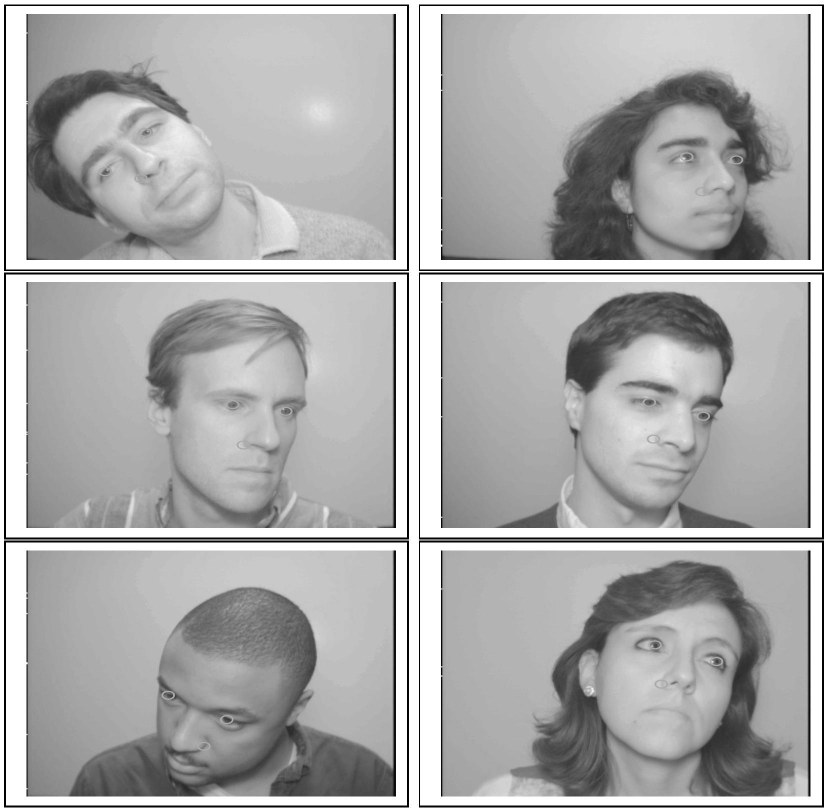


Figure 3-4: Iris and nose lobe features located by the feature finder in some example test images.

To evaluate these feature finder locations, the system was run on all 1550 images in the database, the 15 modeling and 10 testing images of each of the 62 people. For a particular test run, let d_{max} be the maximum distance between a detected feature and its manually chosen location. Four different feature finder outcomes were recorded: good ($d_{max} < t_{good}$), marginal ($t_{good} \leq d_{max} < t_{marginal}$), bad ($d_{max} \geq t_{marginal}$), and null (no features found; all hypotheses rejected). We chose t_{good} to be about 15% of the interocular distance d and $t_{marginal}$ to be 20% of d. In our exhaustive test of the database, the system achieved a good outcome in 99.3% of the images, a marginal outcome in 0.3% of the images, and a bad outcome in 0.4%. No null cases were reported. For the 99.6% of the good and marginal cases, the average distance between the manually and automatically determined feature locations is 1.3 pixels, or about 2% of the interocular distance. The feature locations in either the good or marginal outcomes are sufficient for the geometrical alignment stage of the recognizer, so the recognizer can be run on the vast majority of the test images.

In most of the error cases, the far eye in a rotated face is misplaced, perhaps being located in a nearby dark region such as an eyebrow or a sliver of hair. Even in these cases, however, the nearer eye and the nose are correctly located. In all 1550 database images except one, the feature finder returned at least two good features.

The pose estimated by the system is simply given by the out-of-plane rotation of the best matching level 0 template. In the present system this estimate is not always correct, primarily because the image warping based on optical flow makes matching a little *too* flexible. Sometimes the warping actually changes the pose of the input to match templates from a different pose. Since it is difficult for the warping operation to transform between leftward-looking poses and rightward-looking ones, the pose estimate can reliably distinguish between these two cases. Thus, the pose estimate passed on to the recognizer is currently "looking left" or "looking right". Even though this is a very coarse estimate, since pose estimation is only used to index the example views, we can compensate by simply letting more views get through the indexing stage. Also, it should be possible to place a more refined pose estimation stage after feature extraction, an estimation stage that would use fixed templates and no warping operations.

Because of the large number of templates, the computation takes around 10-15 minutes on a Sun Sparc 2. Using fewer exemplars decreases the running time but also reduces system flexibility and recognition performance.

3.5 Summary

In this chapter, we presented a pose- and person-independent system for automatically locating the two eyes and a nose feature. The system is template-based, employing templates of the eyes and nose region from different "exemplar" people and poses. The problem of feature finding is cast as finding a good match between the input and one of the example templates. A hierarchical coarse-to-fine implementation is described, and the system correctly locates all three features in 99.6% of our 1550 database images.

The next step in our view-based face recognizer is to use the eyes and nose features for geometrical registration in a template-based recognizer, which is the topic of the next chapter.

Chapter 4

Face recognition using multiple views

As mentioned in the introduction, the pictorial representation for face recognition has been quite successful on frontal views of the face, with the template-based approach being a good example (Baron[11], Brunelli and Poggio[28], Gilbert and Yang [59]). In this chapter, our goal is to extend template-based systems to handle varying pose, notably facial rotations in depth. Our approach is view-based, representing faces with templates from many example images that cover the viewing sphere, the 15 views per person shown in Fig. 1-3. In this chapter we describe the view-based recognizer and experimental results when real views are used for the 15 example views per person. The generation and use of *virtual* views for face recognition will be discussed in Chapter 7.

The general outline of our view-based approach to the problem of pose-invariant face recognition is shown in Fig. 4-1. First, in an off-line step, 15 example images are taken of everyone in the database, and templates are extracted and stored to disk. As shown with the thick grey arrows, the on-line procedure uses these templates in the geometrical registration and correlation steps.

To recognize the person in an input image, the system follows the on-line procedure in Fig. 4-1. First, the person- and pose-independent feature finder from Chapter 3 locates the iris and nose lobe features. Based on a coarse pose estimate from the feature finder, the face recognizer next culls the example views, selecting 9 of the 15 views. Then the recognizer loops through the example views of each person, matching each to the input by geometrically registering the two and then correlating example templates against the registered input image. Finally, the recognizer reports the person who has the best set of matching example templates.

We now describe the details of the off-line and on-line procedures, with the exception of the feature finder, which was described in Chapter 3.

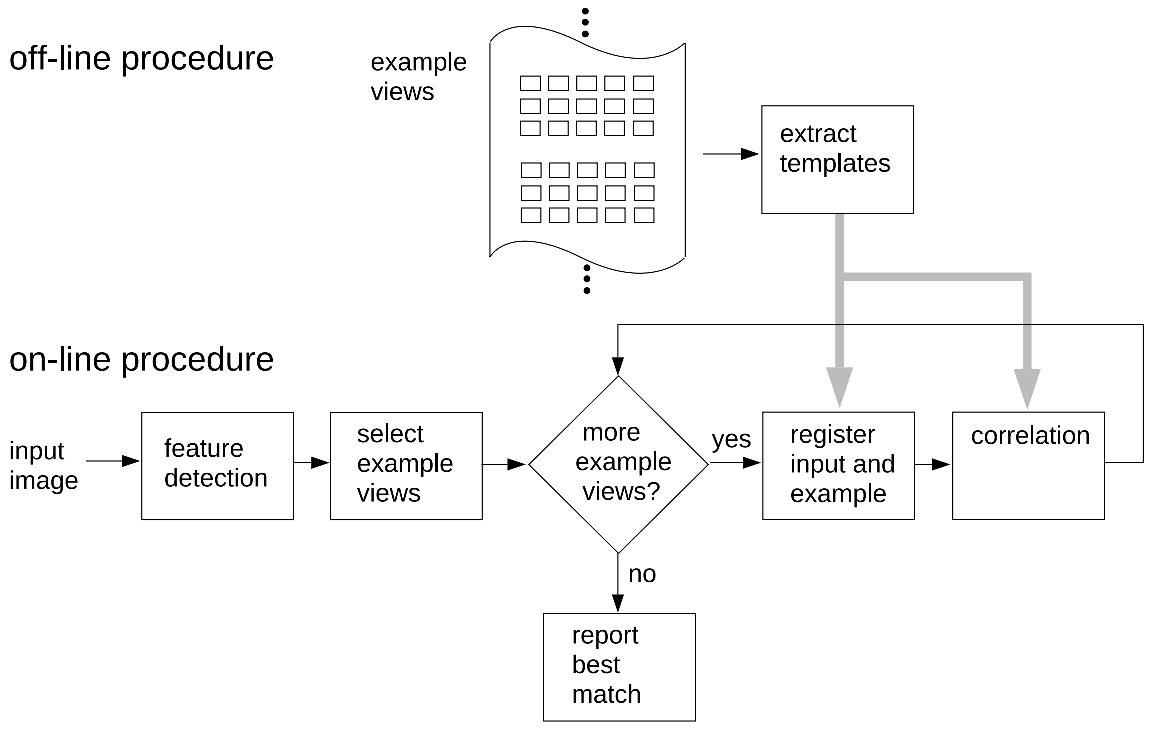


Figure 4-1: The general outline for the off-line and on-line procedures for our view-based approach to pose-invariant face recognition.

4.1 Off-line template extraction

In the off-line preparation of templates, the first step is to take the 15 example images of each person. As discussed in more detail in Appendix A, a uniform set of poses are taken for each person by fixing piece of foam core around the camera, where the desired poses of the 15 views are indicated by dots on the foam core. Each person is asked to rotate their face to point their nose at each one of the dots. After taking the example views, a set of facial features are manually located on the face to center the bounding boxes for the templates. These feature locations, which include the irises, nose lobes, and corners of the mouth, are shown in Fig. 4-2 for an **m3** view and are denoted $\mathbf{p}_i^m, 0 \leq i \leq 5$. The features are located manually since the desired feature set includes more features than are returned by our automatic feature finder.

The next step in preparing the templates is to resample the example images to remove the effects of image-plane rotation and scale. This is done by applying a two point similarity transform to place the irises at a pair of fixed locations $(\mathbf{p}_0, \mathbf{p}_1)$. That is, we solve for a similarity transform T that maps from the destination image img_{sim}

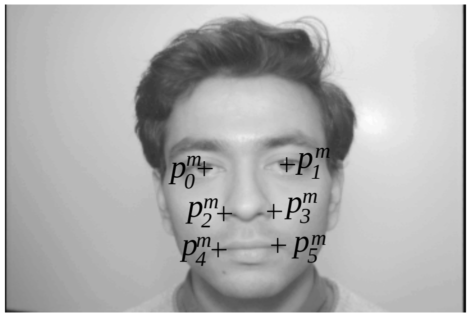


Figure 4-2: To place the template bounding boxes, the irises, nose lobes and corners of the mouth are manually labeled.

to the original example image *img*

$$img_{sim}(\mathbf{p}) = img(T(\mathbf{p})).$$

The direction of this mapping is chosen to facilitate remapping the example image *img* under the similarity transform: a pixel $\mathbf{p}$ in img_{sim} fetches a grey level value from $img(T(\mathbf{p}))$. The similarity transform has the form

$$T(\mathbf{p}) = s \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \mathbf{p} + \begin{pmatrix} t_x \\ t_y \end{pmatrix},$$

where the scale s, image-plane rotation θ, and 2D translation (t_x, t_y) are found by solving

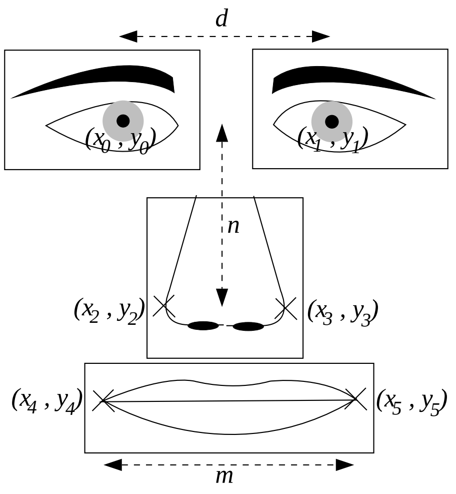
$$T(\mathbf{p}_0) = \mathbf{p}_0^m, \qquad T(\mathbf{p}_1) = \mathbf{p}_1^m.$$

As long as the scale factor s is non-zero, we can invert T to get the transformation from *img* to img_{sim}. Then we can apply T^{-1} to the remaining nose and mouth features, thus defining them in the resampled image img_{sim}

$$\mathbf{p}_i = T^{-1}(\mathbf{p}_i^m), \quad 2 \le i \le 5.$$

The $\mathbf{p}_i$ feature locations are measured relative to a coordinate frame defined by a "whole face template" which will be discussed shortly.

After geometrically normalizing the example images for scale and image-plane rotation, bounding boxes are defined for the eyes, nose, and mouth templates. The placement and sizes of the bounding boxes are determined by the points $\mathbf{p}_i$ and a few geometrical measurements on the face, as shown in Fig. 4-3. The measurements include the interocular distance d, the vertical distance n between the midpoint of



feature	bounding box	
	upper left	lower right
left eye	$(x_0 - .4d,$ $y_0 - .3d)$	$(x_0 + .4d,$ $y_0 + .3d)$
right eye	$(x_1 - .4d,)$ $y_1 - .3d)$	$(x_1 + .4d,$ $y_1 + .3d)$
nose	$(x_2 - .1d,$ $(y_2 + y_3)/2 - .7n)$	$(x_3 + .1d,$ $(y_2 + y_3)/2 + .3n)$
mouth	$(x_4 - .1m,$ $\min(y_4, y_5) - .3m)$	$(x_5 + .1m,$ $\max(y_4, y_5) + .3m)$

Figure 4-3: Given the manually located points $\mathbf{p}_i = (x_i, y_i)$, $0 \leq i \leq 5$, the distances d, n, and m are calculated, and then the bounding boxes for the eyes, nose, and mouth templates are computed. The bounding box formulas are for view **m3**.

the irises and the midpoint of the nose lobes, and the horizontal distance m between the corners of the mouth. The equations for the upper left and lower right corners of the bounding boxes for an **m3** view are shown in the table in Fig. 4-3. The other views use slightly different sets of constant factors for d, n, and m.

Before using the bounding boxes to extract the templates from the example image img_{sim}, we filter img_{sim} with a preprocessing filter. This is to support later experiments that test the face recognizer with different types of grey level preprocessing. The preprocessing filters include the original grey levels I, the gradient magnitude $\|\nabla I\|$, the x and y components of the gradient $\partial_x I$ and $\partial_y I$, and the Laplacian $\partial_{xx} I + \partial_{yy} I$. After preprocessing the image with a filter, the eye, nose, and mouth templates are extracted using the computed bounding boxes. Fig. 4-4 shows some example templates under the various types of preprocessing.

Besides image preprocessing, the overall scale of the templates, as measured by the interocular distance d, is another template design parameter we examined. Scale was varied by changing the distance between the "destination" eye locations $\mathbf{p}_0$ and $\mathbf{p}_1$ in the transformation T. Three different interocular distances d were evaluated: 15, 30, and 60 pixels, with the latter being close to the original image resolution (Fig. 4-5). To avoid problems with aliasing for the 15 and 30 pixel cases, the example image was smoothed before downsampling. The experiments with preprocessing and scale will be described in section 4.3, the experimental results section.

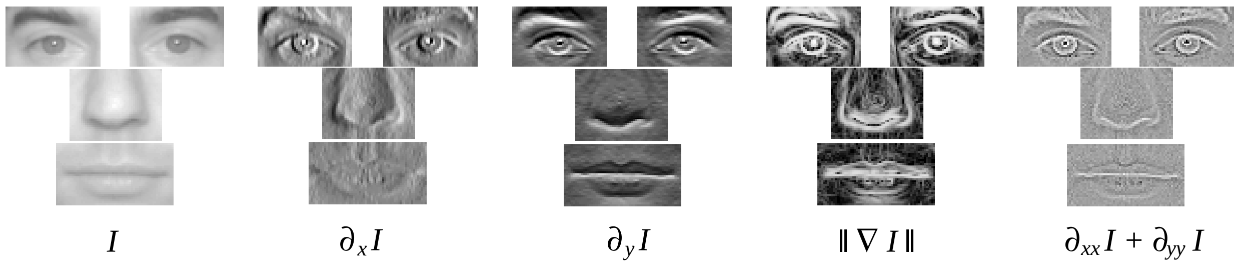


Figure 4-4: Feature templates under different types of preprocessing.

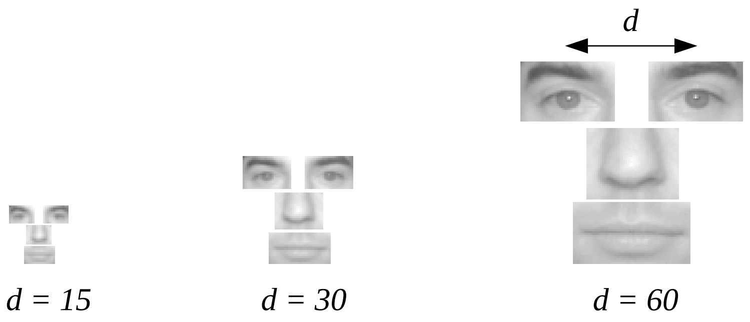


Figure 4-5: Feature templates under different scales, as determined by d, the interocular distance.

At this point, templates for each example image have been created for a variety of image preprocessings and scales. As shown in Fig. 4-6(a), let us denote the individual feature templates as $templ_j, 0 \leq j \leq 3$. These templates form the basis for the correlation step in the on-line recognition procedure in Fig. 4-1. Additional information, however, is stored with the templates for the on-line geometrical registration step and is shown in Fig. 4-6(b). First, to bring the input view into rough correspondence with the example view, the feature points $\mathbf{p}_i$ are stored. To drive the second part of the geometrical registration, a fine, pixelwise correspondence step, a grey level whole face template *face-templ* is stored. The feature locations $\mathbf{p}_i$ are defined relative to this whole face template. The use of the $\mathbf{p}_i$ and *face-templ* will be described in the next section.

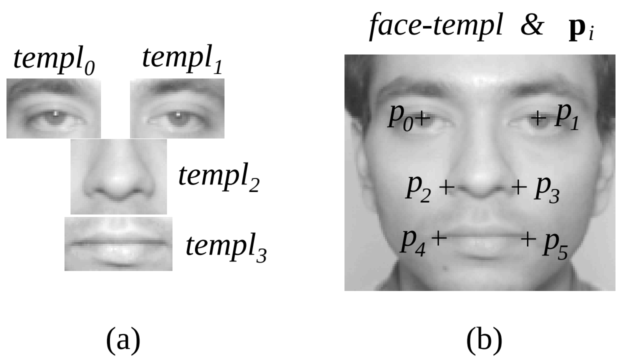


Figure 4-6: The information stored for an example view: (a) templates $templ_j$ of the eyes, nose, and mouth, and (b) information used to assist finding correspondence prior to correlation, a whole face template *face-templ* and a set of feature points $\mathbf{p}_i$.

4.2 On-line recognition algorithm

In this section, we describe the details of our view-based approach to pose-invariant face recognition. Processing in our face recognizer follows the flow diagram for the "on-line" procedure in Fig. 4-1, and pseudocode sketching the steps of our recognizer is given in Fig. 4-7.

Overall, our view-based face recognizer takes as input a view of an unidentified person, compares it against all the people in the database, and returns the best match. The notation for recognizer input, output, and the database is as follows, where person is a person from the database and view is one of the 15 example views.

input a view img_{input} of an unknown person to identify.

database Faces from the database, as described in the previous section, are represented by a set of templates $templ_j$ of the eyes, nose, and mouth regions. In addition, for geometrically registering the input with example views, the feature points $\mathbf{p}_i$ and whole face templates *face-templ* are stored for each view of each person. These templates and feature points are organized in 2D arrays indexed by person and view:

1. Templates. $templ_j[\text{person}][\text{view}], 0 \leq j \leq 3$.
2. Feature locations. $\mathbf{p}_i[\text{person}][\text{view}], 0 \leq i \leq 5$.
3. Whole face template. *face-templ*[person][view]

output the closest matching person in the database.

Template-based recognizer

```
(1) $\mathbf{p}_0^f, \mathbf{p}_1^f, \mathbf{p}_{2|3}^f \leftarrow$ feature finder $(img_{input})$
(2) selected views $\leftarrow$ left or right group of views, from feature finder
(3) for person $\leftarrow$ 1 to NUM_PEOPLE                /* for all people in database */
(4)     forall view $\in$ selected views                  /* for all views to search */
(5)         geometrical registration
                affine transform: $img_{aff}(\mathbf{p}) \leftarrow img_{input}(T(\mathbf{p}))$
                optical flow: $(\mathbf{\Delta x}, \mathbf{\Delta y}) \leftarrow$ optical-flow(face-templ[person][view], $img_{aff}$)
                        $img_{warp}(x,y) \leftarrow img_{aff}(x + \mathbf{\Delta x}(x,y), y + \mathbf{\Delta y}(x,y))$
(6)         correlation
                for $j \leftarrow 0$ to 3                          /* loop over eyes, nose, mouth */
                    cor$_j$[person][view] $\leftarrow$ norm-correlation $(img_{warp}, templ_j$[person][view])
(7)     score[person] $\leftarrow \sum_{j=0}^{3}(\max_{\text{view} \in \text{selected views}}(\text{cor}_j[\text{person}][\text{view}]))$
(8) return $\arg\max_{\text{person}}$ score[person]
```

Figure 4-7: Pseudocode for our template-based recognizer.

To serve as a running example of the algorithm, consider the pair of input and example images shown in Fig. 4-8. Let the example image be view view_{ex} of person $\text{person}_{\text{ex}}$.

Step (1) locates a set of facial features that will be used later in step (5), the geometrical registration step. The feature finder, which is described in Chapter 3, locates the left iris $\mathbf{p}_0^f$, right iris $\mathbf{p}_1^f$, and one of the two nose lobes $\mathbf{p}_{2|3}^f$. The notation 2|3 means that either the left or right nose lobe feature is returned, which depends on the left/right rotation of the input. Fig. 4-9(a) shows the features detected for the input image in Fig. 4-8.

The left vs. right rotation information provided by the feature finder is used in step (2) to filter the example views. The left vs. right distinction, while quite coarse, provides an estimate of the out-of-plane rotation of the input. This can be used as a filter on the example views: only those example views that are similar in view direction to the input will be selected. The views selected by the recognizer are either the left three columns or right three columns of Fig. 1-3

$$\begin{aligned} \text{selected-views (left)} &= \{\mathbf{m10}, \mathbf{m9}, \mathbf{m8}, \mathbf{m5}, \mathbf{m4}, \mathbf{m3}, \mathbf{m15}, \mathbf{m14}, \mathbf{m13}\} \\ \text{selected-views (right)} &= \{\mathbf{m8}, \mathbf{m7}, \mathbf{m6}, \mathbf{m3}, \mathbf{m2}, \mathbf{m1}, \mathbf{m13}, \mathbf{m12}, \mathbf{m11}\}. \end{aligned}$$

Ideally, one would want a more refined estimate of out-of-plane rotation, which would allow the recognizer to further winnow down the number of example views it needs

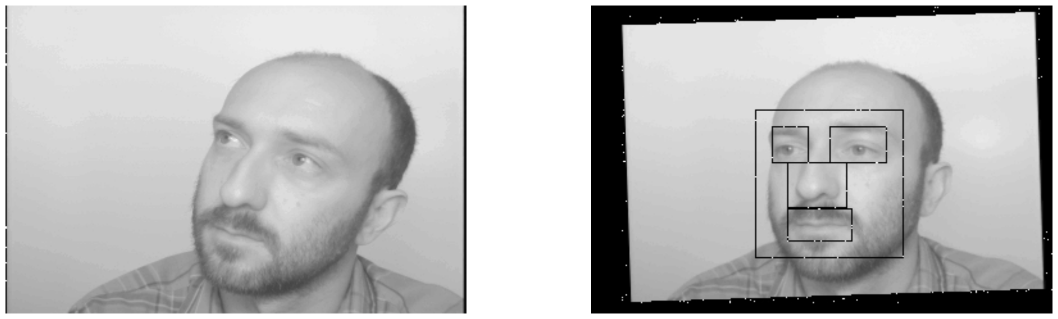


Figure 4-8: We demonstrate the recognition algorithm matching the example input image (left) with the example view (right). The example view is view $\mathsf{view}_{\mathsf{ex}}$ of person $\mathsf{person}_{\mathsf{ex}}$.

to test for each person.

Next, in steps (3) and (4) the recognizer loops over the selected example views of all people, matching each in turn against the input. To record the template correlation scores for the selected example views, a multidimensional array $\mathsf{cor}_j[\cdot][\cdot]$ is established, which parallels $templ_j[\cdot][\cdot]$ in being indexed by feature index j, person, and view.

The main part of the recognizer, steps (5) and (6), compares the input image against a particular example view. This comparison consists of geometrical registration (step (5)) followed by correlation (step (6)). The geometrical alignment step brings the input and example images into close spatial correspondence in preparation for the correlation step. To geometrically align the input image against the example image, first an affine transform is applied to the input to align three feature points, the two eyes and a nose lobe feature

$$img_{aff}(\mathbf{p}) = img_{input}(T(\mathbf{p})).$$

The affine transform has the form

$$T(\mathbf{p}) = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \mathbf{p} + \begin{pmatrix} t_x \\ t_y \end{pmatrix},$$

where the affine parameters a_{00}, a_{01}, a_{10}, a_{11}, t_x, and t_y are found by solving

$$T(\mathbf{p}_0) = \mathbf{p}_0^f, \qquad T(\mathbf{p}_1) = \mathbf{p}_1^f, \qquad T(\mathbf{p}_{2|3}) = \mathbf{p}_{2|3}^f.$$

This three point affine transform essentially models the face as a planar object passing through the three feature points. This transformation can correctly compensate for the 2D aspects of pose: scale, image-plane rotation, and 2D translation. For rotations

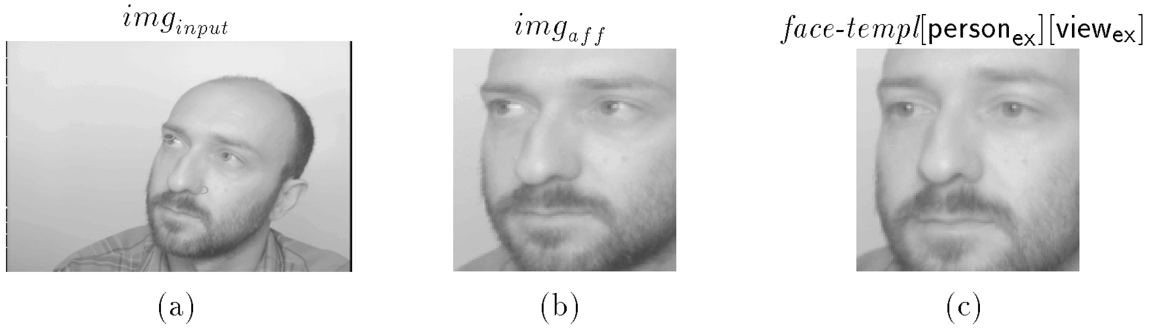

img_{input}
img_{aff}
$\textit{face-templ}[\mathsf{person}_{\mathsf{ex}}][\mathsf{view}_{\mathsf{ex}}]$
(a)
(b)
(c)

Figure 4-9: The features detected in the input image (a) are used to affine transform the input (b) so that the features brought into correspondence with the same features in the example view (c).

in depth, the accuracy of the compensation for a given point on the face deteriorates the further the point is from the plane defined by the feature points. For the interior portion of the face, this typically affects the tip of the nose more than the other features. Fig. 4-9(b) shows the result of affine transforming the input image (a) to align its features with those of the example image in (c).

The second part of the geometrical alignment step attempts to compensate for any small remaining geometrical differences between the affine transformed input img_{aff} (Fig. 4-9(b)) and the whole face template $\textit{face-templ}[\mathsf{person}_{\mathsf{ex}}][\mathsf{view}_{\mathsf{ex}}]$ (Fig. 4-9(c)). These remaining differences may be due to factors such as out-of-plane rotation, expression, gaze direction, or errors in the feature detection module. A dense set of pixelwise correspondences between the affine transformed input and the whole face template is computed using a hierarchical, gradient-based optical flow algorithm [18]

$$(\mathbf{\Delta x}, \mathbf{\Delta y}) = \text{optical-flow}(\textit{face-templ}[\mathsf{person}_{\mathsf{ex}}][\mathsf{view}_{\mathsf{ex}}], img_{aff}).$$

The vector field $(\mathbf{\Delta x}, \mathbf{\Delta y})$ specifies for each pixel (x, y) in the whole face template a relative offset $(\mathbf{\Delta x}(x, y), \mathbf{\Delta y}(x, y))$ to the corresponding pixel in the affine transformed input. Thus, by applying a 2D warp operation driven by the optical flow, the affine transformed input can be brought into pixel-level correspondence with the whole face template

$$img_{warp}(x, y) = img_{aff}(x + \mathbf{\Delta x}(x, y), y + \mathbf{\Delta y}(x, y)).$$

Basically, pixels in the affine transformed input are "pushed" along the flow vectors to their corresponding pixels in the whole face template. Fig. 4-10 shows the pixelwise correspondence process for the input and example images of Fig. 4-8.

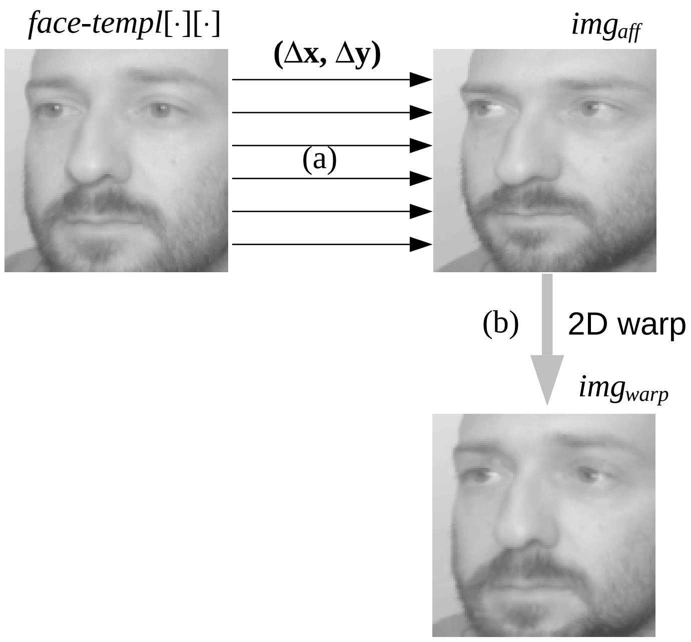


Figure 4-10: The pixelwise correspondence part of the geometrical registration step: (a) pixelwise correspondences are computed between *face-templ*[person$_{ex}$][view$_{ex}$] and img_{aff} using an optical flow algorithm, and (b) the correspondences are used to drive a 2D warp of img_{aff} to produce img_{warp}. Images *face-templ*[person$_{ex}$][view$_{ex}$] and img_{warp} are now in pixelwise correspondence.

When the input and example are the same person, optical flow generally succeeds in finding correspondence and can compensate for small rotation, scale, and expression differences between the affine transformed input and example image. When the input and example are from different people, optical flow can fail to find correct correspondence, in which case the 2D warp distorts the image and the eventual template match will be poor. This failure case, however, does not matter since we want to reject the match anyway.

Overall, this two-stage geometrical registration technique of aligning a set of points followed by optical flow is related to the topic of affine shape (see Koenderink and van Doorn [79] and Shashua [119] [120]).

Now that the input and example images have been geometrically registered, in step (6) the eye, nose, and mouth templates from the example image are correlated against the input img_{warp}. First, to match the preprocessing used for the example templates, the input img_{warp} is filtered using the same preprocessing filter. The different preprocessing filters were described in the previous section on template extraction. Next, each example template is correlated over a small region (e.g. 5x5) centered around its expected location in img_{warp}. Normalized correlation is the matching metric

$$r = \frac{< \mathbf{TI} > - < \mathbf{T} >< \mathbf{I} >}{\sigma(\mathbf{T})\sigma(\mathbf{I})},$$

where $\mathbf{T}$ is the template, $\mathbf{I}$ is the subportion of image being matched against, $<>$ is the mean operator, and $\sigma()$ measures standard deviation. As mentioned in the introduction, normalized correlation provides an invariance to grey level shifts of the template $\mathbf{T}$ of the form $a\mathbf{T} + b$, where a is a constant scaling factor and b is an additive constant. This kind of invariance may provide immunity to differences between template and image in the overall ambient lighting level or camera contrast.

When scoring a person in step (7), the system takes the sum of correlations from the best matching eye, nose, and mouth templates. Note that we maximize over the views separately for each template, so the best matching left eye could be from view 1 and the best matching nose from view 2, and so on. We found that switching the order of the sum and max operations – first summing template scores and then maximizing over views – gives slightly worse performance, probably because the original sum/max ordering is more flexible.

After comparing the input against all people in the database, the recognizer in step (8) returns the person with the highest correlation score – we have not yet developed a criterion on how good a match has to be to be believable. A first step in studying this problem could be to compare the correlation score statistics for correct matches against those for incorrect matches. Considering a task like face verification, having the ability to reject inputs is important and is something we plan under future work.

4.3 Experimental results

Our view-based face recognizer was evaluated on a test set of 620 images. Described in the introduction and Appendix A, the test set contains 10 views each of 62 people. When taking the test set, subjects in the database were asked to present their face at a series of random poses to the camera, where the pose is constrained to lie within the overall range of example view poses. In addition, the subjects are encouraged to rotate their face in the image plane for half the images, so all three rotation parameters are varied in the test set. Example sets of test images are shown in Figs. 1-6 and A-2.

On this test set our face recognizer basically achieves a recognition rate of 98%. To explore the effects of changing the template scale and preprocessing filter on this recognition rate, we have performed a series of recognition experiments, where each experiment runs through the entire test set of 620 images. The recognition

preprocessing	performance – 620 test images								bad features	
	correct		2nd place		3rd place		>3rd place			
dx+dy	98.71%	(612)	0.32%	(2)	0.48%	(3)	0.16%	(1)	0.32%	(2)
mag	98.23%	(609)	0.81%	(5)	0.32%	(2)	0.32%	(2)	0.32%	(2)
lap	98.07%	(608)	0.81%	(5)	0.32%	(2)	0.48%	(3)	0.32%	(2)
grey	94.52%	(586)	1.94%	(12)	0.48%	(3)	2.74%	(17)	0.32%	(2)

Table 4.1: Face recognition performance versus preprocessing. Best performance is from using the gradient magnitude (mag), Laplacian (lap), or the sum of separate correlations on the x and y gradient components (dx+dy). An intermediate scale was used, with an interocular distance of 30.

rate in these experiments counts errors in *both* the feature finder and template-based recognition. That is, if the feature finder fails, then the template-based recognizer is not executed and an error is recorded. Our feature finder failed to find the eyes and nose locations in two test images, so the experiments with preprocessing and template scale begin with a handicap of 2 images. These error cases are listed in the rightmost column of tables 4.1 and 4.2.

Table 4.1 summarizes our recognition results for the preprocessing experiments. The types of preprocessing we tested include the gradient magnitude (mag), Laplacian (lap), sum of separate correlations on x and y components of the gradient (dx+dy), and the original grey levels (grey). For these preprocessing experiments we used an intermediate template scale, an interocular distance of 30. In table 4.1, we list the number of correct recognitions and the number of times the correct person came in second, third, or past third place. Best performance was had from dx+dy, mag, and lap, with dx+dy yielding the best recognition rate at 98.7%. Preprocessing with the gradient magnitude performs nearly as well, a result in agreement with the preprocessing experiments of Brunelli and Poggio[28]. Given that using the original grey levels produces the lower rate of 94.5%, our results indicate that preprocessing the image with a differential operator gives the system a performance advantage. We think the performance differences between dx+dy, mag, and lap are too small to say that one preprocessing type stands out over the others.

Table 4.2 summarizes our recognition results for the template scale experiments, where scale is measured by the interocular distance of a frontal view. The preprocessing was fixed at dx+dy. The intermediate and fine scales perform the best, indicating that at least for our input representation, the coarsest scale may be losing detail needed to distinguish between people. Since the intermediate scale has a computa-

interocular distance	performance – 620 test images				
	correct	2nd place	3rd place	>3rd place	bad features
15	96.13% (596)	2.26% (14)	0.32% (2)	0.97% (6)	0.32% (2)
30	98.71% (612)	0.32% (2)	0.48% (3)	0.16% (1)	0.32% (2)
60	98.39% (610)	0.81% (5)	0.16% (1)	0.32% (2)	0.32% (2)

Table 4.2: Face recognition performance versus scale, as measured by interocular distance (in pixels). The intermediate scale performs the best, a result in agreement with Brunelli and Poggio[28]. For preprocessing, separate correlations on the x and y components of the gradient were computed and then summed (dx+dy).

tional advantage over the finer scale, we would recommend operating a face recognizer at the intermediate scale.

One additional experimental question to ask is how necessary optical flow is to the geometrical registration step. To evaluate the impact of optical flow in the recognizer, we removed optical flow and tested the recognizer under the different types of preprocessing. Fig. 4-11 shows the result, where the light bars indicate the original result using optical flow, and the dark colored bars without optical flow. Template scale for this experiment was fixed at an interocular distance of 30 pixels. As evident from the bar graph, excluding optical flow results in a drop in the recognition rate of roughly 3% for the original grey levels to 10% for the Laplacian. This difference between the Laplacian and the grey levels may be due to the fact that the Laplacian uses higher frequency information, which may make it more susceptible to slight misregistrations. Overall, these experiments show that the optical flow step does indeed improve our view-based recognizer.

Getting back to the results from tables 4.1 and 4.2, consider the errors made for the best combination of preprocessing and scale: dx+dy at an intermediate scale. Of the 8 errors, 2 were due to the feature finder and 6 were recognition errors. In the one recognition error where the correct person was not even among the top three, the correspondences from optical flow were poor. For the other errors, the correct person came in either second or third place. For these false positive matches, using optical flow to warp the input to the model may be contributing to the problem. If two people are similar enough, the optical flow can effectively "morph" one person into the other, making the matcher a bit *too* flexible at times.

The problem with optical flow sometimes making the matcher too flexible suggests some extensions to the recognizer. Since we only want to compensate for rotational, scale, or expression changes and not allow "identity-changing" transforms, perhaps

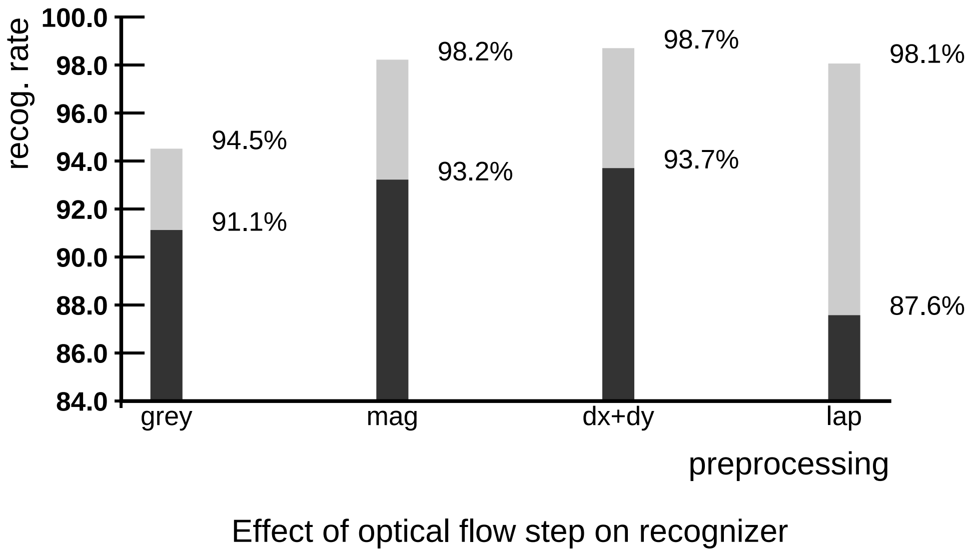


Figure 4-11: How much does the recognition rate drop when the optical flow step is removed? In this figure, we compare the recognition rates with optical flow (light bars) to the recognition rates without optical flow (dark bars) for the different types of preprocessing. Thus, optical flow does have an noticeable impact on the registration step.

the optical flow can be interpreted and the match discarded if the optical flow is not from the allowed class of transformations. Another approach would be to penalize a match using some smoothness measure of optical flow. The new matching metric would have a regularized flavor, being the sum of correlation and smoothness terms

$$\|I(x + \mathbf{\Delta x}(x, y), y + \mathbf{\Delta y}(x, y)) - T\|^2 + \lambda\phi(\mathbf{\Delta x}, \mathbf{\Delta y}),$$

where $I(x + \mathbf{\Delta x}(x, y), y + \mathbf{\Delta y}(x, y))$ is the input warped by the flow $(\mathbf{\Delta x}, \mathbf{\Delta y})$, T is the template, ϕ is a smoothness functional including derivatives, and λ is a parameter controlling the trade off between correlation and smoothness. This functional has an interpretation as the combination of a noise model on the intensity image and priors on the flow.

Another way to constrain the flow-based matching procedure would be to introduce a model for the types of allowable deformations and replace the optical flow routine with a "model-based matching" routine. One possible model for deformations is a linear combination of example deformations

$$(\mathbf{\Delta x}, \mathbf{\Delta y}) = \sum_{i=1}^{n} \alpha_i(\mathbf{\Delta x}_i, \mathbf{\Delta y}_i),$$

where the $(\mathbf{\Delta x}_i, \mathbf{\Delta y}_i)$ are example deformations such as small expression, scale, and rotation changes. This is related to the shape models of Cootes, *et al.* [44], Blake and Isard [21], Baumberg and Hogg [12], and Jones and Poggio [72]. The example deformations may be captured, for instance, by collecting images of a prototype face undergoing those types of transformations. Model-based matching constrains the matching task because aligning and image I with a template T now involves solving for α_i that satisfy

$$I(x + \sum_{i=1}^{n} \alpha_i \mathbf{\Delta x}_i(x,y), y + \sum_{i=1}^{n} \alpha_i \mathbf{\Delta y}_i(x,y)) = T.$$

This equation only has n unknown parameters, probably on the order of 10 or 20. This is opposed to the optical flow calculation, which has twice the number of pixels as unknowns. Ideally, with this kind of model for deformations, nonsensical or identity-transforming deformations can be avoided, but this remains to be demonstrated for the recognition task.

Besides adding constraints on the flow-based correspondences, another technique for increasing the overall discrimination power of the face representation would be to add information about face geometry. A geometrical feature vector of distances and angles that is similar to current feature geometry approaches could be tried, but the representation would have to be extended to deal with varying pose.

In terms of execution time, our current system takes about 1 second to do each input/model comparison on a Sun Sparc 1. The computation time is dominated by resampling the image during the affine transform, optical flow, and correlation. On our unoptimized CM-5 implementation, it takes about 10 seconds for the template-based recognizer to run since we can distribute the database so that each processor compares the input against one person. Specialized hardware, for example correlation chips[59], can be used to further speed up the computation.

4.4 Summary

In this chapter we presented a view-based approach for recognizing faces under varying pose. Motivated by the success of recent template-based approaches for frontal views, our approach models faces with templates from 15 views that sample different poses from the viewing sphere. The recognizer consists of two main stages, a geometrical alignment stage where the input is registered with the model views and a correlation stage for matching. Our recognizer has achieved a recognition rate of 98% on a database 62 people. The database consists of 930 model views and 620 testing views covering a variety of poses, including rotations in depth and rotations in the image plane.

In the first part of the thesis, we have looked at the problem of pose-invariant face recognition when multiple views of each database person are available. The view-based system described in this chapter has shown that template-based face recognition systems can be extended in a straightforward way to handle the problem of varying pose. But what if only one view is available of each person in the database? Is pose-invariant face recognition still possible? This is the topic of the second half of the thesis.

Part II

Face Recognition Using Virtual Views

Chapter 5

A vectorized image representation

In the first half of the thesis, our view-based face recognizer used templates to represent faces, a representation that proved to be sufficient for the matching task faced by the recognizer. However, the second half of the thesis, virtual views, places a heavier burden on our face representation. Our example-based techniques for generating virtual views use a *vectorized* face representation, which is an ordered vector of image measurements taken at a set of facial feature points. These features can run the gamut from sparse features with semantic meaning, such as the corners of the eyes and mouth, to pixel level features that are defined by the local grey level structure of the image. By an ordered vector, we mean that the facial features have been enumerated $f_1, f_2, \ldots, f_n$, and that the vector representation first contains measurements from f_1, then f_2, etc. The measurements at a given feature will include its (x, y) location – a measure of face "shape" – and local image color or intensity – a measure of face "texture". The key part of this vectorized representation is that the facial features $f_1, f_2, \ldots, f_n$ are effectively put into correspondence across the face images being "vectorized". For example, if f_1 is the outer corner of the left eye, then the first three elements of our vector representation will refer to the $(x_1, y_1, \text{intensity-patch}(x_1, y_1))$ measurements of that feature point for any face being vectorized.

Establishing feature correspondence among a set of face images is important for our techniques for synthesizing virtual views of a novel face. Once corresponding features have been found for a set of face images, it makes sense to speak of things like taking linear combinations of faces or computing a geometrical distortion between two face shapes. For example, for the idea of linear classes, the space of face shapes and textures is modeled using linear combinations of the prototype faces. In the technique of parallel deformation, the vectorized shape component can be used to create a geometric mapping between the prototype face and the novel face. The use of the vectorized representation for virtual views will be discussed in Chapter 7.

Computing the vectorized representation is essentially a feature detection or correspondence finding task. The difficulty of the correspondence task depends of the set of face images being vectorized, and here we distinguish between the two cases of *inter*person and *intra*person correspondence. In the former case of interperson correspondence, the set of face images being vectorized contains images of different people. For our purposes, the face images will be at the same pose-expression-lighting parameters, so the main difficulty is handling the variability in facial appearance seen across different people. That is, the set of feature correspondences need to be computed even when faces appear quite different due to differences in race, age, gender, facial hair, etc. This is distinguished from the simpler problem of *intra*person correspondence, which involves finding correspondence between images of the same person. As opposed to the interperson correspondence case, the images here will differ by a slight rotation or expression. For this problem, a relatively simple correspondence algorithm such as optical flow is usually sufficient for locating corresponding features.

In this chapter, we introduce notation for the vectorized image representation and overview techniques for computing the correspondences required to drive the representation. Probably on the order of tens of feature correspondences need to be located to sufficiently characterize face shape, so a technique more advanced than our feature finder of Chapter 3 is required. This chapter discusses three correspondence techniques, a manual approach by Beier and Neely [13], optical flow, and a novel approach that we call an "image vectorizer". The last approach is described in detail in Chapter 6. The vectorized notation and correspondence techniques are discussed here in preparation for the virtual views synthesis techniques of Chapter 7, which draw heavily on this chapter and the next.

Before moving on, it is important to note that the necessity for feature correspondence in synthesizing virtual views is not an artifact of using the example-based approach. In a general setting, one can consider our use of vectorization within virtual views as "registration" of the novel view with our prior knowledge of faces. In the competing approach of using a generic 3D models for prior knowledge, one faces the similar task of finding correspondence between points in the image data and the 3D model. As mentioned in Chapter 2 on previous work, this registration task is usually accomplished either by acquiring the 3D range data and image data simultaneously using specialized equipment such as the Cyberware scanner, or corresponding points on the image and 3D model are automatically or manually determined.

5.1 Vectorized shape and texture

As previously mentioned, there are two components in the vectorized image representation, face shape and texture. The first component, face shape, is a measure of the locations of facial features. The second component, face texture, is a measure of color or intensity values at the feature points defining face shape. These two components will be represented and processed as separate vectors. In this section, we introduce notation and discuss the computation of the vectorized shape and texture components.

5.1.1 Shape

Given the locations of features $f_1, f_2, \ldots, f_n$, shape is represented by a vector $\mathbf{y}$ of length $2n$ consisting of the concatenation of the x and y coordinate values

$$\mathbf{y} = \begin{pmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{pmatrix}.$$

In our notation, if an image being vectorized has an identifying subscript (e.g. i_a), then the vector $\mathbf{y}$ will carry the same subscript, $\mathbf{y}_a$. The coordinate system used for measuring x and y will be one normalized by using the eye locations to fix interocular distance and remove head tilt. By factoring out the 2D aspects of pose, the remaining variability in shape vectors will be caused by expressions, rotations out of the image plane, and the natural variation in the configuration of features seen across people.

This vectorized representation for 2D shape has been widely used, including network-based object recognition (Poggio and Edelman [107]), the linear combinations approach to recognition (Ullman and Basri [130], Poggio [105]), active shape models (Cootes and Taylor [42], Cootes, *et al.* [44]) and face recognition (Craw and Cameron [45][46]). In these shape vectors, a sparse set of feature points, on the order of 10's of features, are either manually placed on the object or located using a feature finder. For a face, example feature points may include the inner and outer corners of the eyes, the corners of the mouth, and points along the eyebrows and sides of the face.

In this thesis we use a dense representation of one feature per pixel, a representation originally suggested to us by the object recognition work of Shashua [119]. Compared to a sparser representation, the pixelwise representation increases the difficulty of finding correspondences. However, we have found that a standard optical flow algorithm [18], preceded by normalization based on the eye locations, can do a good job at automatically computing dense pixelwise correspondences. After defining one image as a "reference" image, the (x, y) locations of feature points of a new image are computed by finding optical flow between the two images. Thus the shape

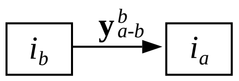


Figure 5-1: In relative shape, $\mathbf{y}^b_{a-b}$ denotes feature correspondence between i_b and i_a using i_b as a reference.

vector of the new image, really a "relative" shape, is described by a flow or a vector field of correspondences relative to a standard reference shape. Our face vectorizer from Chapter 6, which uses optical flow as a subroutine, is also used to automatically compute the vectorized representation.

Optical flow matches features in the two frames using the local grey level structure of the images. As opposed to a feature finder, where the "semantics" of features is determined in advance by the particular set of features sought by the feature finder, the reference image provides shape "semantics" in the relative representation. For example, to find the corner of the left eye in a relative shape, one follows the vector field starting from the left eye corner pixel in the reference image.

Correspondence with respect to a reference shape, as computed by optical flow, can be expressed in our vector notation as the difference between two vectorized shapes. Let us chose a face shape $\mathbf{y}_b$ to be the reference. Then the shape of an arbitrary face $\mathbf{y}_a$ is represented by the geometrical difference $\mathbf{y}_a - \mathbf{y}_b$, which we shall abbreviate $\mathbf{y}_{a-b}$. This is still a vector of length $2n$, but now it is a vector field of correspondences between images i_a and i_b. In addition, we keep track of the reference frame by using a superscript, so we add the superscript b to the shape $\mathbf{y}^b_{a-b}$. The utility of keeping track of the reference image will become more apparent when describing operations on shapes. Pictorially, we visualize the shape $\mathbf{y}^b_{a-b}$ in Fig. 5-1 by drawing an arrow from i_b to i_a. This relative shape representation has been used by Beymer, Shashua, and Poggio [19] in an example-based approach to image analysis and synthesis.

5.1.2 Texture

Given a set of features $f_1, f_2, \ldots, f_n$ driving an image vectorization, the texture vector $\mathbf{t}$ is a sampling of image intensity or color patches at the feature points. A key point in the texture vector is that features are registered across all faces being vectorized; a given offset in the texture vector $\mathbf{t}[i]$ contains an intensity or color value from the same point f_i on all faces.

Given an image i_a to vectorize, previous work in vectorizing textures has used two methods for representing the texture vector $\mathbf{t}_a$. First, a "feature-based" method forms $\mathbf{t}_a$ out of small patches of intensities $\mathbf{t}_{a,i}$ centered around the features

$$\mathbf{t}_a = \begin{pmatrix} \mathbf{t}_{a,1} \\ \mathbf{t}_{a,2} \\ \vdots \\ \mathbf{t}_{a,n} \end{pmatrix},$$

where the vector $\mathbf{t}_{a,i}$ is some function of the local image patch $i_a(x + x_i, y + y_i)$. The template-based approach to face recognition (e.g. Baron [11], Brunelli and Poggio [28], Bichsel [20]) can be seen as a very coarse vectorization where the function $\mathbf{t}_{a,i}$ is a lexicographical scan of the patch. In the active shape models of Cootes and Taylor [43], feature points are grouped along boundaries, and the vector $\mathbf{t}_{a,i}$ is a 1D set of grey-levels sampled along a line perpendicular to the boundary. In the face recognition work of Manjunath, *et al.* [91], the function $\mathbf{t}_{a,i}$ is a set of filter responses to Gabor filters of differing scales and orientations centered at feature point f_i. The spatial extent of the local patch depends on the density of the features, with sparser features using larger patches. A pixelwise shape representation would only require a patch consisting of one pixel.

The second textural representation creates a geometrically normalized version of the image i_a. That is, the geometrical differences among face images are factored out by warping the images to a common reference shape. This strategy for representing texture has been used, for example, in the face recognition works of Craw and Cameron [45], and Shackleton and Welsh [118]. If we let shape $\mathbf{y}_{std}$ be the reference shape, then the geometrically normalized image $\mathbf{t}_a$ is given by the 2D warp

$$\mathbf{t}_a(x, y) = i_a(x + \mathbf{\Delta x}^{std}_{a-std}(x, y), y + \mathbf{\Delta y}^{std}_{a-std}(x, y)),$$

where $\mathbf{\Delta x}^{std}_{a-std}$ and $\mathbf{\Delta y}^{std}_{a-std}$ are the x and y components of the pixelwise mapping between $\mathbf{y}_a$ and the standard shape $\mathbf{y}_{std}$. These pixelwise correspondences are derived from the shape components $\mathbf{y}_a$ and $\mathbf{y}_{std}$, or basically the relative shape $\mathbf{y}^{std}_{a-std}$. If shape is sparsely defined, then texture mapping or sparse data interpolation techniques can be employed to create the necessary pixelwise level representation. Example sparse data interpolation techniques include using splines (Litwinowicz and Williams [87], Wolberg [138]), radial basis functions (Reisfeld, Arad, and Yeshurun [114]), and inverse weighted distance metrics (Beier and Neely [13]). If a pixelwise representation is being used for shape in the first place, such as one derived from optical flow, then texture mapping or data interpolation techniques can be avoided.

For a pixelwise shape representation, the two approaches converge, with the feature-based approach essentially becoming geometrical normalization. That is, each vector $\mathbf{t}_{a,i}$ is really just one pixel, so the entire collection of n pixels can be viewed as an image if the pixels are arranged in 2D using a reference face shape. This points to the advantage of using a dense, pixelwise representation. Texture processing is simplified over the sparse case since we avoid texture mapping and sparse data interpolation techniques, instead employing a simple 2D warping algorithm. Additionally, though, using a pixelwise representation makes the vectorized representation very simple conceptually: we can think of three measurements being made per feature $(x, y, I(x, y))$. The price we pay for this simplicity is a difficult correspondence problem. In the next section we describe three correspondence techniques we explored for computing the vectorized image representation.

5.2 Computing the vectorized representation

As mentioned previously, when vectorizing a group of images the correspondence problem breaks down into two subcases of differing difficulty, interperson correspondence and intraperson correspondence. The former is more difficult than the latter since the former must handle the variation in facial appearance seen across different people while the latter deals with images of just one person. Since both of these cases are encountered in virtual views, we have investigated three correspondence techniques: optical flow for intraperson correspondence, and for interperson correspondence, a manual method and a novel automatic technique that we call an image vectorizer. This section briefly covers these three techniques, and Chapter 6 provides a more thorough description of our image vectorizer.

The pixelwise correspondence algorithms discussed in this section compute a relative shape $\mathbf{y}^b_{a-b}$, i.e. the shape $\mathbf{y}_a$ of image i_a with respect to a reference image i_b. This computation will be denoted using the **vect** operator

$$\mathbf{y}^b_{a-b} = \mathbf{vect}(i_a, i_b).$$

Of course, given this relative shape $\mathbf{y}^b_{a-b}$, our original "absolute" definition of shape $\mathbf{y}^b_a$ can be computed by simply adding the shape $\mathbf{y}^b_b$, which is simply the x and y coordinate values of each pixel in i_b. This **vect** operator notation will be used later in Chapter 7 on synthesizing virtual views.

5.2.1 A manual approach

The manual correspondence technique we used for interperson correspondence was borrowed from Beier and Neely's morphing technique in computer graphics [13]. In their technique, a dense correspondence map between two images is created by applying a sparse data interpolation technique to a set of manual feature correspondences.

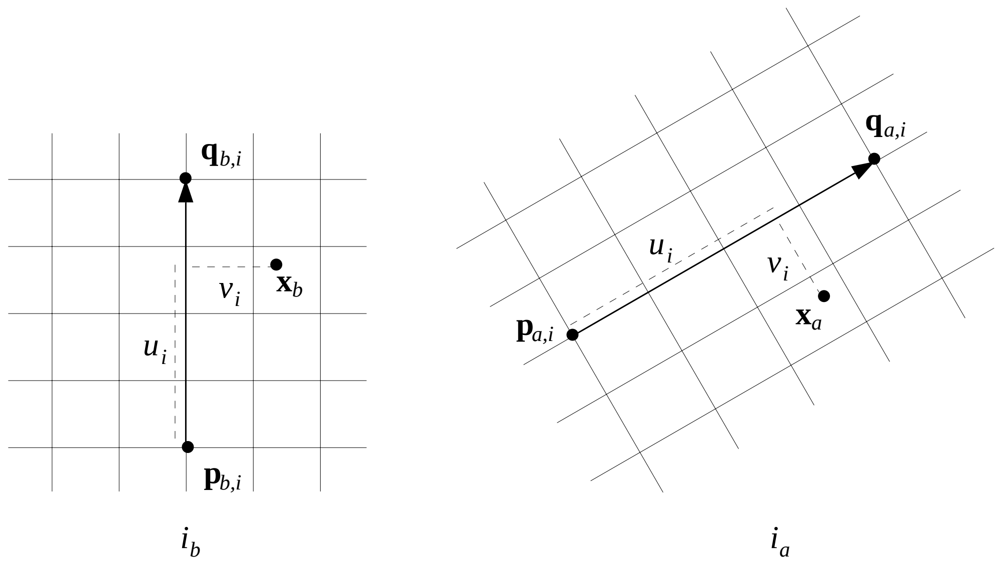


Figure 5-2: The correspondence technique of Beier and Neely interpolates the displacements of corresponding line segment features. Here we want to compute $\mathbf{x}_a$ given the point $\mathbf{x}_b$. Figure after Beier and Neely [13].

The image features are line segments, and the generated correspondence map interpolates the displacements between corresponding line segment features.

Consider the displacement field created by a single pair of segment features. As shown in Fig. 5-2, let the ith pair of segment features be $\overline{\mathbf{p}_{a,i}\mathbf{q}_{a,i}}$ from image i_a and $\overline{\mathbf{p}_{b,i}\mathbf{q}_{b,i}}$ from image i_b. The displacement field, indicated by the two grids, resembles a similarity transform except that there is no scaling perpendicular to the segment, just scaling along it. To compute the contribution of this displacement field to the correspondence field $\mathbf{y}^b_{a-b}$, we need to compute the point $\mathbf{x}_a$ from its corresponding point $\mathbf{x}_b$. This places the reference frame of the correspondence field in image i_b, allowing the correspondences to be used in a geometrically normalizing transform from image i_a to the reference image i_b. To compute the location $\mathbf{x}_a$, first in the image i_b we compute the coordinates (u_i, v_i) of $\mathbf{x}_b$ with respect to the segment $\overline{\mathbf{p}_{b,i}\mathbf{q}_{b,i}}$

$$\begin{aligned} u_i &= \frac{(\mathbf{x}_b - \mathbf{p}_{b,i}) \cdot (\mathbf{q}_{b,i} - \mathbf{p}_{b,i})}{\|\mathbf{q}_{b,i} - \mathbf{p}_{b,i}\|^2} \\ v_i &= \frac{(\mathbf{x}_b - \mathbf{p}_{b,i}) \cdot \text{Perpendicular}(\mathbf{q}_{b,i} - \mathbf{p}_{b,i})}{\|\mathbf{q}_{b,i} - \mathbf{p}_{b,i}\|}. \end{aligned}$$

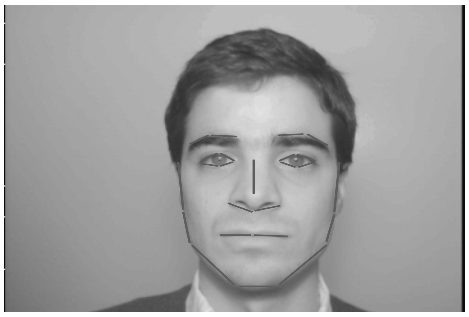


Figure 5-3: Manually entered line segments driving correspondence for the sparse data interpolation method of Beier and Neely [13].

The point $\mathbf{x}_a$ in image i_a is then placed relative to the line segment $\overline{\mathbf{p}_{a,i}\mathbf{q}_{a,i}}$

$$\mathbf{x}_a = \mathbf{p}_{a,i} + u_i(\mathbf{q}_{a,i} - \mathbf{p}_{a,i}) + \frac{v_i \text{ Perpendicular}(\mathbf{q}_{a,i} - \mathbf{p}_{a,i})}{\|(\mathbf{q}_{a,i} - \mathbf{p}_{a,i})\|},$$

so the computed displacement at $\mathbf{x}_b$ due to the ith line segment pairing is

$$\boldsymbol{\Delta}_i(\mathbf{x}_b) = \mathbf{x}_a - \mathbf{x}_b.$$

The displacement field $\boldsymbol{\Delta}_i$ is computed for a set of n segment correspondences $(1 \leq i \leq n)$; the feature set we use for a frontal view is shown in Fig. 5-3.

The final correspondence field $\mathbf{y}^b_{a-b}$ is a weighted average of the n displacements $\boldsymbol{\Delta}_i$

$$\mathbf{y}^b_{a-b}(\mathbf{x}_b) = \frac{\sum_{i=1}^n w_i(\mathbf{x}_b)\boldsymbol{\Delta}_i(\mathbf{x}_b)}{\sum_{i=1}^n w_i(\mathbf{x}_b)},$$

where the weighting function $w_i(\mathbf{x}_b)$ is inversely proportional to the distance from $\mathbf{x}_b$ to the segment $\overline{\mathbf{p}_{b,i}\mathbf{q}_{b,i}}$

$$\begin{aligned} w_i(\mathbf{x}_b) &= \left(\frac{\|\mathbf{q}_{b,i} - \mathbf{p}_{b,i}\|^{c_1}}{c_2 + \text{dist}_i(\mathbf{x}_b)}\right)^{c_3} \quad \text{for } c_1,\, c_2,\, c_3 \text{ constants} \\ \text{dist}_i(\mathbf{x}_b) &= \begin{cases} |v_i|, & 0 \leq u_i \leq 1 \\ \|\mathbf{x}_b - \mathbf{p}_{b,i}\|, & u_i < 0 \\ \|\mathbf{x}_b - \mathbf{q}_{b,i}\|, & u_i > 1. \end{cases} \end{aligned}$$

The weighting function also favors longer segments over shorter ones by incorporating the length of segment $\overline{\mathbf{p}_{b,i}\mathbf{q}_{b,i}}$. We have used constant values of $c_1 = 0.5$, $c_2 = 2.0$, and $c_3 = 1.5$ (as suggested by Beier and Neely [13]). Overall, the weighting makes the final correspondence field a sum of n local transforms, each one exerting an influence in the regions surrounding $\overline{\mathbf{p}_{b,i}\mathbf{q}_{b,i}}$ in i_b and $\overline{\mathbf{p}_{a,i}\mathbf{q}_{a,i}}$ in i_a.

5.2.2 Optical flow

Optical flow originates from a subfield of computer vision called motion vision. Given a sequence of image frames containing moving objects and taken from an observer that is perhaps moving itself, the goal of motion vision is to interpret the motion of scene objects and the observer. While one class of motion methods, known as "direct" methods, estimate object motion directly from the images and their spatiotemporal derivatives, another popular class of motion methods begins with the computation of a low level, image-based measurement called optical flow. Optical flow, a quantity defined on two successive image frames, is an estimate of the local translations between corresponding grey-level intensity patches in the two frames. These low-level correspondences are typically represented as a vector mapping between pixels in images at time t and $t+1$.

The basic mathematical assumption underlying the computation of optical flow is that the intensity value of a moving point in the scene does not change between two frames. Following the development in Horn and Schunk [68], if we consider the image sequence to be a function $I(x, y, t)$ of spatial locations x and y and time t, then the brightness constraint can be expressed at a point (x, y, t) as

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t). \tag{5.1}$$

Expanding the left hand side using the Taylor expansion

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} + \text{higher order terms},$$

neglecting the higher order terms, and plugging back into equation (5.1) yields

$$\delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} = 0.$$

If we divide through by δt and make the substitutions $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}, u = \frac{\delta x}{\delta t}, v = \frac{\delta x}{\delta t}$, then we get

$$u\ I_x + v\ I_y + I_t = 0. \tag{5.2}$$

In this equation for point (x, y), the optical flow (u, v) are the unknowns, and measurements I_x, I_y, and I_t are the computed spatial and temporal derivatives of the function $I(x, y, t)$. The fact that this equation is underconstrained – two unknowns and one equation – leads to a problem known as the "aperture problem". This is typically addressed by introducing a spatial smoothness constraint to regularize the solution of optical flow.

While many algorithms exist for estimating optical flow, we have used a hierarchical, gradient-based algorithm (Lucas and Kanade [90], Bergen and Adelson [16], Bergen and Hingorani [18], Bergen, *et al.* [17]). Since factors such as noise makes an exact solution of equation (5.2) at a point (x, y) impractical, a solution is found by minimizing the quadratic error

$$\min(u\ I_x + v\ I_y + I_t)^2.$$

To implement the smoothness constraint, we assume that (u, v) is constant over a small image region R centered around (x, y) (e.g. 5x5 pixels) and thus integrate the squared error term over a region R

$$\min \sum_R (u\ I_x + v\ I_y + I_t)^2.$$

Differentiating with respect to u and v and setting each partial derivative to zero yields the 2x2 system

$$\begin{bmatrix} \sum_R I_x^2 & \sum_R I_x I_y \\ \sum_R I_x I_y & \sum_R I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum_R I_x I_t \\ -\sum_R I_y I_t \end{bmatrix}.$$

To solve for this system robustly, one must look at the rank of the matrix

$$\begin{bmatrix} \sum_R I_x^2 & \sum_R I_x I_y \\ \sum_R I_x I_y & \sum_R I_y^2 \end{bmatrix},$$

which can be estimated by inspecting its eigenvalues. Bergen and Hingorani [18] describe a robust solution based on the magnitudes of the eigenvalues. In addition, the confidence of the optical flow (u, v) can be estimated by the magnitude of the smaller eigenvalue.

Since the brightness constraint equation (5.2) is derived from a local Taylor series expansion of the image function $I(x, y, t)$, this optical flow algorithm is only designed to handle small pixel displacements, on the order of a single pixel. In order to efficiently handle optical flow correspondences across multiple pixel displacements, the above algorithm is applied in a hierarchical coarse-to-fine approach. First, the Laplacian pyramids (see Burt and Adelson [33]) of the two images are formed, and processing begins at the coarsest level. Reducing the image to a coarse resolution makes large displacements at the original image resolution smaller at the reduced scale, allowing the flow algorithm to find correct correspondences at the reduced scale. As processing moves to the next finer scale, the flow from the previous level is used to compensate for the motion between the two images by warping the first image to bring it closer to the second image. The remaining differences between the warped first image and the second image should be on the order of a pixel. Thus, the residual flow can be computed using the above gradient-based technique and added to the flow estimate from the previous level.

Getting back to the original problem of computing vectorized shape, we can apply the optical flow algorithm to the problem of finding intraperson correspondence. One image of the person is chosen as a "reference" image, and other images are vectorized with respect to it by computing optical flow. To assist the correspondence process, the two images are registered to compensate for any differences in 2D translation, scale, and image-plane rotation. This registration is performed using a similarity transform to align the eyes of the image being vectorized with those of the reference image. The eye locations, specified by the center of the irises, can be automatically or manually located.

A key limitation with this technique, however, is that the optical flow algorithm will fail to find correct correspondences when the two faces are dissimilar enough in appearance. For example, finding interperson correspondence often breaks down when the two people are from different races. Even with intraperson correspondence, correspondence will fail if the images are separated by a large enough rotation in depth. Thus, when covering an out-of-plane rotation of more than several degrees, we use a sequence of images at intermediate poses to assist the correspondence process; this is described in more detail in Chapter 7.

5.2.3 Face vectorizer

Our face vectorizer, to be described in detail in Chapter 6, is an automatic technique for computing both the shape and texture components of our vectorized representation. While optical flow is used as a subroutine in the vectorizer, the vectorizer is capable of handling the difficult case of interperson correspondence that sometimes foils optical flow. What makes the face vectorizer superior to optical flow alone is the explicit modeling of the texture component. Texture is modeled using linear combinations of example textures as in the eigenimage approach to face recognition (Turk and Pentland [129], Pentland, *et al.* [103]).

Here we briefly sketch the process of vectorizing the shape and texture of an image i_a. Shape is measured by finding pixelwise correspondence $\mathbf{y}_{a-std}^{std}$ relative to a standard face shape. This is done by finding the optical flow between i_a and a "reference" image at standard face shape, an image that is produced from the texture model. Standard face shape will be defined as the average of many example prototype face shapes. Face texture is modeled using the assumption that the space of face images is linearly spanned by a prototypical set of example face images. That is, the texture of i_a is modeled by taking a linear combination of a set of n geometrically normalized example prototype textures $\mathbf{t}_{p_j}, 1 \leq j \leq n$

$$\mathbf{t}_a = \sum_{j=1}^{n} \beta_j \mathbf{t}_{p_j}. \tag{5.3}$$

The β_j coefficients are computed by projecting a geometrically normalized version of i_a onto the space spanned by the $\mathbf{t}_{p_j}$.

The key to our vectorization procedure is to link the computation steps of shape and texture so that each one depends on the other. Thus, improving the estimate of one improves the other, and we can iterate back and forth between shape and texture steps until the vectorized representation converges. For example, the shape $\mathbf{y}^{std}_{a-std}$ is used in the texture step to geometrically normalize the input i_a before projecting onto the texture examples $\mathbf{t}_{p_j}$

$$i_{warp} = i_a(x + \mathbf{\Delta x}^{std}_{a-std}(x,y), y + \mathbf{\Delta y}^{std}_{a-std}(x,y)),$$

where $\mathbf{\Delta x}^{std}_{a-std}$ and $\mathbf{\Delta y}^{std}_{a-std}$ are the x and y components of $\mathbf{y}^{std}_{a-std}$. This assists the texture step by aligning the facial features in i_{warp} with the features in the geometrically normalized textures. Going in the other direction, the texture computation assists shape by reconstructing a geometrically normalized grey level version of the input (equation (5.3)). This synthesized "reference" image can be used to compute shape correspondences using a simple algorithm like optical flow. Overall, the vectorization procedure iteratively solves for a flow $(\mathbf{\Delta x}^{std}_{a-std}, \mathbf{\Delta y}^{std}_{a-std})$ and coefficients β_j that solve

$$i_a(x + \mathbf{\Delta x}^{std}_{a-std}(x,y), y + \mathbf{\Delta y}^{std}_{a-std}(x,y)) = \sum_{j=1}^{n} \beta_j t_{p_j}.$$

Correspondence between two arbitrary images can thus be found by vectorizing both, as now both images are in correspondence with the average shape. After vectorizing both images, one flow is "inverted", and the two flows are then "concatenated". Shape operations such as inversion and concatenation are the subject of the next section.

5.3 Warping and shape manipulation operators

In this section we describe operators on the vectorized representation that will be useful in describing our methods for synthesizing virtual views in Chapter 7.

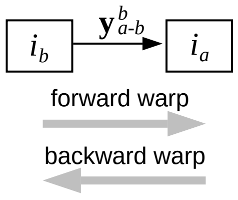


Figure 5-4: A forward warp moves pixels from i_b to i_a using the shape $\mathbf{y}^b_{a-b}$. A backward warp is the inverse, moving pixels from i_a to i_b.

5.3.1 Warping operators

2D warping operations locally distort the arrangement of pixel intensities in an image by following the correspondence vectors in a relative shape vector $\mathbf{y}^b_{a-b}$. Using nomenclature from the computer graphics community, we define two types of warping operations, backward and forward warping (Fig. 5-4). The difference between the two lies in the direction that pixel values travel between shapes $\mathbf{y}_a$ and $\mathbf{y}_b$. A backward warp "retrieves" pixels from an image at shape $\mathbf{y}_a$ and places them at their corresponding locations in reference shape $\mathbf{y}_b$. Inversely, a forward warp "pushes" pixels in an image at shape $\mathbf{y}_b$ forward along the correspondence vectors to the shape $\mathbf{y}_a$.

Backward warp

In an example backward warp, let i_a be an arbitrary image at shape $\mathbf{y}_a$ that we wish to warp to the shape $\mathbf{y}_b$, producing i_b. Since the correspondences $\mathbf{y}^b_{a-b}$ are defined relative to $\mathbf{y}_b$, pixels in i_b can simply use the correspondence field to "index" into image i_a

$$i_b(\mathbf{q}_b) = i_a(\mathbf{q}_b + \mathbf{y}^b_{a-b}(\mathbf{q}_b)), \tag{5.4}$$

where $\mathbf{q}_b$ is a 2D pixel location in $\mathbf{y}_b$ and $\mathbf{q}_a = \mathbf{q}_b + \mathbf{y}^b_{a-b}(\mathbf{q}_b)$ is the corresponding point in $\mathbf{y}_a$. Since in general $\mathbf{q}_a$ will not be at an integral pixel location in $\mathbf{y}_a$, bilinear interpolation is used to sample an intensity value from i_a. As shown in Fig. 5-5, we interpolate i_a at the set of four pixels $\{\mathbf{p}_{a,i}\}^3_{i=0}$ neighboring $\mathbf{q}_a$

$$i_b(\mathbf{q}_b) = \sum_{i=0}^{3} w_i(\alpha,\beta)\, i_a(\mathbf{p}_{a,i}),$$

where

$$(\alpha,\beta) = \mathbf{q}_a - \mathbf{p}_{a,0}$$

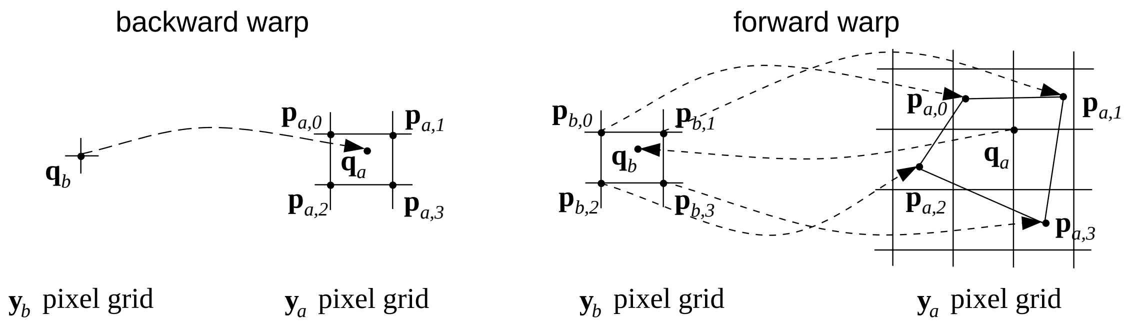


Figure 5-5: Interpolation for backward and forward warping. In a backward warp, point $\mathbf{q}_b$ retrieves a value from $\mathbf{q}_a$. In a forward warp, first the correspondences $\mathbf{y}^b_{a-b}$ are inverted to produce $\mathbf{y}^a_{b-a}$, as depicted by the arrow from $\mathbf{q}_a$ to $\mathbf{q}_b$. Then point $\mathbf{q}_a$ retrieves a value from $\mathbf{q}_b$ as in a backward warp. See text for more details.

and

$$\begin{array}{ll} w_0(\alpha,\beta) = (1-\alpha)(1-\beta) & w_1(\alpha,\beta) = \alpha(1-\beta) \\ w_2(\alpha,\beta) = (1-\alpha)\beta & w_3(\alpha,\beta) = \alpha\beta. \end{array} \tag{5.5}$$

When refering to backward warping in the future chapters, we will either use the mathematical form in equation (5.4) or the notation $i_b = \mathbf{bwarp}(i_a, \mathbf{y}^b_{a-b})$.

Forward warp

Given an arbitrary image i_b at shape $\mathbf{y}_b$, a forward warp sends a pixel value $i_b(\mathbf{q}_b)$ to the point $\mathbf{q}_a$, its corresponding location in shape $\mathbf{y}_a$, where $\mathbf{q}_a = \mathbf{q}_b + \mathbf{y}^b_{a-b}(\mathbf{q}_b)$. Since the point $\mathbf{q}_a$ may not be at an integral location, it is not immediately clear how to store the grey level value in i_a, the warped version of i_b. We approach this problem by first inverting the shape $\mathbf{y}^b_{a-b}$ to produce $\mathbf{y}^a_{b-a}$. By reversing the directions of the correspondence field, the forward warping thus becomes a backward warping, so the previously described backward warping algorithm can be applied. The key part is inverting the correspondences. Referring to Fig. 5-5, our goal is to construct the correspondence from a pixel $\mathbf{q}_a$ to $\mathbf{q}_b$ given that the correspondences are defined in opposite direction.

Inverting the correspondences $\mathbf{y}^b_{a-b}$ is solved using the idea of four corner mapping (see Wolberg[138]). We repeat the following steps for every square source patch of four adjacent pixels $\{\mathbf{p}_{b,i}\}_{i=0}^3$ in i_b. Map the source patch to a quadrilateral $\{\mathbf{p}_{a,i}\}_{i=0}^3$ in the shape $\mathbf{y}_a$

$$\mathbf{p}_{a,i} = \mathbf{q}_{b,i} + \mathbf{y}^b_{a-b}(\mathbf{q}_{b,i}), \quad 0 \le i \le 3.$$

For each pixel $\mathbf{q}_a$ inside this quadrilateral, we estimate its position inside the quadrilateral treating the sides of the quadrilateral as a warped coordinate system. The parametric position (α, β) within the quadrilateral is estimated by solving the 2x2 nonlinear system for bilinear interpolation

$$\mathbf{q}_a = \sum_{i=0}^{3} w_i(\alpha, \beta)\mathbf{p}_{a,i}$$

using a Newton-Raphson method (see section 9.6 of [112]), where the w_i are as defined in equation (5.5). Note that the (α, β) position lies within the unit square. This position is then used to map to a location $\mathbf{q}_b$ in the original source patch

$$\mathbf{q}_b = \mathbf{p}_{b,0} + (\alpha, \beta)$$

and the correspondence field $\mathbf{y}^a_{b-a}$ has been determined at a point

$$\mathbf{y}^a_{b-a}(\mathbf{q}_a) = \mathbf{q}_b - \mathbf{q}_a.$$

Processing now proceeds as with a backward warping.

The notation **fwarp** will be used in Chapter 7 to denote the forward warping procedure. Forward warping image i_b to i_a will be written as $i_a = \mathbf{fwarp}(i_b, \mathbf{y}^b_{a-b})$. In general, we can push pixels along any arbitrary flow x, yielding the more general form of $i_{b+x} = \mathbf{fwarp}(i_b, \mathbf{y}^b_x)$. The only restriction is that the subscript of the image argument must match the superscript of the shape argument, implying that the image must be in the reference frame of the shape.

5.3.2 Shape manipulation operators

In this section, we introduce shape manipulation operators that generate new shapes from shape arguments.

First, shapes can be combined using binary operations such as addition and subtraction. In adding and subtracting shapes, the reference frames of both shapes must be the same, and the subscripts of the shape arguments are added/subtracted to yield the subscripts of the results: $\mathbf{y}^b_{u\pm v} = \mathbf{y}^b_u \pm \mathbf{y}^b_v$.

The reference frame of a shape $\mathbf{y}^b_x$ can be changed from i_b to i_a by applying a forward warp with the shape $\mathbf{y}^b_{a-b}$. Shown pictorially in Fig. 5-6(a), the operation consists of separate 2D forward warps on the x and y components of $\mathbf{y}^b_x$ interpreted for the moment as images instead of vectors. Instead of pushing grey level pixels in the forward warp, we push the x and y components of the shape. The operation in Fig. 5-6(a) is denoted $\mathbf{y}^a_x = \mathbf{fwarp\text{-}vect}(\mathbf{y}^b_x, \mathbf{y}^b_{a-b})$. The inverse operation, shown in

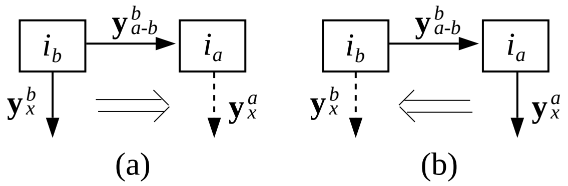
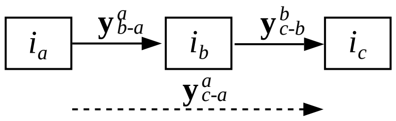


Figure 5-6: (a) To change the reference frame of flow $\mathbf{y}^b_x$ from i_a to i_b, the x and y components are forward warped along $\mathbf{y}^b_{a-b}$, producing the dotted flow $\mathbf{y}^a_x$. In (b), backward warping is used to compute the inverse.



i_a
$\mathbf{y}^a_{b\text{-}a}$
i_b
$\mathbf{y}^b_{c\text{-}b}$
i_c
$\mathbf{y}^a_{c\text{-}a}$

Figure 5-7: In flow concatenation, the flows $\mathbf{y}^a_{b-a}$ and $\mathbf{y}^b_{c-b}$ are composed to produce the dotted flow $\mathbf{y}^a_{c-a}$.

Fig. 5-6(b), is computed using two backward warps instead of forward ones: $\mathbf{y}^b_x = \textbf{bwarp-vect}(\mathbf{y}^a_x, \mathbf{y}^b_{a-b})$.

Finally, two flows fields $\mathbf{y}^a_{b-a}$ and $\mathbf{y}^b_{c-b}$ can be concatenated or composed to produce pixelwise correspondences between i_a and i_c, $\mathbf{y}^a_{c-a}$. Concatenation is shown pictorially in Fig. 5-7 and is denoted $\mathbf{y}^a_{c-a} = \textbf{concat}(\mathbf{y}^a_{b-a}, \mathbf{y}^b_{c-b})$. The basic idea behind implementing this operator is to put both shapes in the same reference frame and then add. This is done by first computing $\mathbf{y}^a_{c-b} = \textbf{bwarp-vect}(\mathbf{y}^b_{c-b}, \mathbf{y}^a_{b-a})$ followed by $\mathbf{y}^a_{c-a} = \mathbf{y}^a_{b-a} + \mathbf{y}^a_{c-b}$.

5.4 Summary

In this chapter, we first introduced a *vectorized* image representation, a feature-based representation where correspondence has been established with respect to a reference image. Two image measurements are made at the feature points. First, feature geometry, or shape, is represented by the (x, y) feature locations relative to some standard face shape. Second, grey levels, or texture, is represented by mapping image grey levels onto the standard face shape.

Next, we discussed three methods for computing this vectorized representation for face images. The problem is basically one of finding feature correspondence with respect to the standard reference shape. The first technique was the sparse data interpolation method of Beier and Neely [13], which relies on a set of manually placed features. Second, optical flow can automatically find correspondences when the two face images are not too far separated in pose, lighting, expression, etc. Finally, our face vectorizer is a novel automatic approach that outperforms optical flow by virtue of incorporating an appearance-based model for face grey levels. The vectorizer is described in more detail in the next chapter.

Finally, the chapter closed with a discussion of backwards and forwards warping operators as well as miscellaneous shape operators such as addition and concatenation. These operators will be useful in Chapter 7 on synthesizing virtual views.

Chapter 6

Vectorizing face images

The previous chapter defined a **vectorized representation** to be a feature-based representation where correspondence has been established relative to a fixed reference object or reference image. In this chapter, we introduce an algorithm for computing the vectorized representation for faces. Computing the vectorized representation can be thought of as arranging the feature sets into ordered vectors so that the ith element of each vector refers to the same feature point for all objects. Given the correspondences in the vectorized representation, applications such as feature detection and pose and expression estimation are possible.

As mentioned in the previous chapter, the two primary components of the vectorized representation are shape and texture. Previous approaches in analyzing faces have stressed either one component or the other, such as feature localization or decomposing texture as a linear combination of eigenfaces (see Turk and Pentland [129]). The key aspect of our vectorization algorithm, or "vectorizer", is that the two processes for the analysis of shape and texture are coupled. That is, the shape and texture processes are coupled by making each process use the output of the other. The texture analysis uses shape for geometrical normalization, and shape analysis uses texture to synthesize a reference image for feature correspondence. Empirically, we have found that this links the two processes in a positive feedback loop. Iterating between the shape and texture steps causes the vectorized representation to converge after several iterations.

Our vectorizer is similar to the active shape model of Cootes, *et al.* [44] [43] in that both iteratively fit a shape/texture model to the input. But there are interesting differences in the modeling of both shape and texture. In our vectorizer there is no model for shape; it is measured in a data-driven manner using optical flow. In active shape models, shape is modeled using a parametric, example-based method. First, an ensemble of shapes are processed using principal component analysis, which produces a set of "eigenshapes". New shapes are then written as linear combinations of these eigenshapes. Texture modeling in their approach, however, is weaker than in

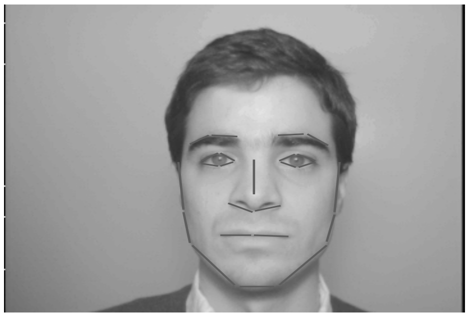


Figure 6-1: To define the shape of the prototypes off-line, manual line segment features are used. After Beier and Neely [13].

ours. Texture is only modeled locally along 1D contours at each of the feature points defining shape. Our approach models texture over larger regions – such as eyes, nose, and mouth templates – which should provide more constraint for textural analysis. In the future we intend to add a model for shape similar to active shape models, as discussed ahead in section 6.6.2.

In this chapter, we first explore the description of the vectorized representation in more detail, focusing on the definition of standard shape. Then the basic coupling of the shape and texture computations is motivated, followed by a description of the vectorization algorithm. A hierarchical coarse-to-fine implementation is described, an application to feature detection is presented, and we close with a discussion of future work.

6.1 Standard shape

Since the vectorized representation is relative to a 2D reference, first we define a standard feature geometry for the reference image. The features on new faces will then be measured relative to the standard geometry. In this chapter, the standard geometry for frontal views of faces is defined by averaging a set of line segment features over an ensemble of "prototype" faces. Fig. 6-1 shows the line segment features for a particular individual, and Fig. 6-2 shows the average over a set of 14 prototype people. Features are assigned a text label (e.g. "c_1") so that corresponding line segments can be paired across images. As we will explain later in section 6.3.1, the line segment features are specified manually in an initial off-line step that defines the standard feature geometry.

The two components of the vectorized representation, shape and texture, can now

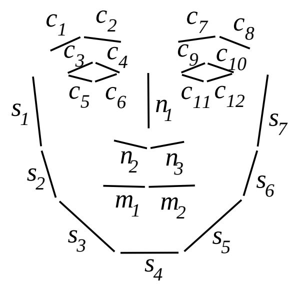


Figure 6-2: Manually defined shapes are averaged to compute the standard face shape.

be defined relative to this standard shape.

6.1.1 Shape

The shape vector of an image i_a, denoted $\mathbf{y}^{std}_{a-std}$, could be sparsely defined, perhaps recording (x, y) locations of i_a's segment features relative to the standard shape in Fig. 6-2. However, to facilitate shape and texture operators in the run-time vectorization procedure, shape is spatially oversampled. That is, we use a pixelwise representation for shape, defining a feature point at each pixel in a subimage containing the face. The shape vector $\mathbf{y}^{std}_{a-std}$ can then be visualized as a vector field of correspondences between a face at standard shape and the given image i_a being represented. If there are n pixels in the face subimage being vectorized, then the shape vector consists of $2n$ values, a $(\delta x, \delta y)$ pair for each pixel. Fig. 6-3 shows the shape representation $\mathbf{y}^{std}_{a-std}$ for the image i_a. As indicated by the grey arrow, correspondences are measured relative to the reference face i_{std} at standard shape. (Image i_{std} in this case is mean grey level image; modeling grey level texture is discussed more in section 6.3.1.) Overall, the advantage of using a dense representation is that it allows a simple optical flow calculation to be used for computing shape and a simple 2D warping operator for geometrical normalization.

6.1.2 Texture

As mentioned in the previous chapter, our texture vector is the grey level image i_a that has been warped onto standard shape. If we let shape $\mathbf{y}_{std}$ be the reference shape, then the geometrically normalized image $\mathbf{t}_a$ is given by the 2D warp

$$\mathbf{t}_a(x, y) = i_a(x + \mathbf{\Delta x}^{std}_{a-std}(x, y), y + \mathbf{\Delta y}^{std}_{a-std}(x, y)),$$

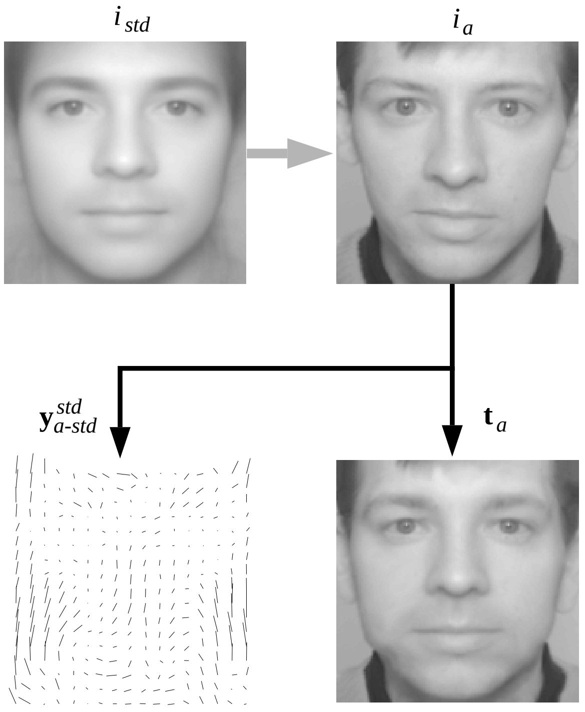


Figure 6-3: Our vectorized representation for image i_a with respect to the reference image i_{std} at standard shape. First, pixelwise correspondence is computed between i_{std} and i_a, as indicated by the grey arrow. Shape $\mathbf{y}^{std}_{a-std}$ is a vector field that specifies a corresponding pixel in i_a for each pixel in i_{std}. Texture $\mathbf{t}_a$ consists of the grey levels of i_a mapped onto the standard shape.

where $\mathbf{\Delta x}^{std}_{a-std}$ and $\mathbf{\Delta y}^{std}_{a-std}$ are the x and y components of $\mathbf{y}^{std}_{a-std}$. Fig. 6-3 in the lower right shows an example texture vector $\mathbf{t}_a$ for the input image i_a in the upper right.

6.1.3 Separation of shape and texture

How cleanly have we separated the notions of shape and texture in the 2D representations just described? Ideally, the ultimate shape description would be a 3D one where the (x, y, z) coordinates are represented. Texture would be a description of local surface albedo at each feature point on the object. Such descriptions are common for the modeling of 3D objects for computer graphics, and it would be nice for vision algorithms to invert the imaging or "rendering" process from 3D models to 2D images.

What our 2D vectorized description has done, however, is to factor out and explicitly represent the salient aspects of 2D shape. The true spatial density of this 2D representation depends, of course, on the density of features defining standard shape, shown in our case in Fig. 6-2. Some aspects of 2D shape, such as lip or eyebrow thickness, will end up being encoded in our model for texture. However, one could extend the standard feature set to include more features around the mouth and eyebrows if desired. For texture, there are non-albedo factors confounded in the texture component, such as lighting conditions and the z-component of shape. Overall, though, remember that only one view of the object being vectorized is available, thus limiting our access to 3D information. We hope that the current definitions of shape and texture are a reasonable approximation to the desired decomposition.

6.2 Shape/texture coupling

One of the main results of this chapter is that the computations for the shape and texture components can be algorithmically coupled. That is, shape can be used to geometrically normalize the input image prior to texture analysis. Likewise, the result of texture analysis can be used to synthesize a reference image for finding correspondences in the shape computation. The result is an iterative algorithm for vectorizing images of faces. Let us now explore the coupling of shape and texture in more detail.

6.2.1 Shape perspective

Since the vectorized representation is determined by an ordered set of feature points, computing the representation is essentially a feature finding or correspondence task. Consider this correspondence task under a special set of circumstances: we know who the person is, and we have prior example views of that person. In this case, a simple correspondence finding algorithm such as optical flow should suffice. As shown in the left two images of Fig. 6-4, first a prior example i_a of the person's face is manually warped in an off-line step to standard shape, producing a reference image $\mathbf{t}_a$. A new image of the same person can now be vectorized simply by running an optical flow algorithm between the image and reference $\mathbf{t}_a$.

If we have no prior knowledge of the person being vectorized, the correspondence problem becomes more difficult. In order to handle the variability seen in facial appearance across different people, one could imagine using many different example reference images that have been pre-warped to the standard reference shape. These reference images could be chosen, for example, by running a clustering algorithm

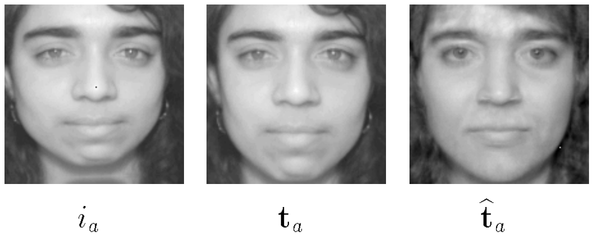


Figure 6-4: Vectorizing face images: if we know who the person is and have prior example views i_a of their face, then we can manually warp i_a to standard shape, producing a reference $\mathbf{t}_a$. New images of the person can be vectorized by computing optical flow between $\mathbf{t}_a$ and the new input. However, if we do not have prior knowledge of the person being vectorized, we can still synthesize an approximation to $\mathbf{t}_a$, $\widehat{\mathbf{t}}_a$, by taking a linear combination of prototype textures.

on a large ensemble of example face images. This solution, however, introduces the problem of having to choose among the reference images for the final vectorization, perhaps based on a confidence measure in the correspondence algorithm.

Going one step further, in this chapter we use a statistical model for facial texture in order to assist the correspondence process. Our texture model relies on the assumption, commonly made in the eigenface approach to face recognition and detection (Turk and Pentland [129], Pentland, *et al.* [103]), that the space of grey level images of faces is linearly spanned by a set of example views. That is, the geometrically normalized texture vector $\mathbf{t}_a$ from the input image i_a can be approximated as a linear combination of n prototype textures $\mathbf{t}_{p_j}, 1 \leq j \leq n$

$$\widehat{\mathbf{t}}_a = \sum_{j=1}^{n} \beta_j \mathbf{t}_{p_j}, \tag{6.1}$$

where the $\mathbf{t}_{p_j}$ are themselves geometrically normalized by warping them to the standard reference shape. The rightmost image of Fig. 6-4, for example, shows an approximation $\widehat{\mathbf{t}}_a$ that is generated by taking a linear combination of textures as in equation (6.1). If the vectorization procedure can estimate a proper set of β_j coefficients, then computing correspondences should be simple. Since the computed "reference" image $\widehat{\mathbf{t}}_a$ approximates the texture $\mathbf{t}_a$ of the input and is geometrically normalized, we are back to the situation where a simple correspondence algorithm like optical flow should work. In addition, the linear β_j coefficients act as a low dimensional code for representing the texture vector $\mathbf{t}_a$.

This raises the question of computing the β_j coefficients for the texture model. Let us now consider the vectorization procedure from the perspective of modeling texture.

6.2.2 Texture perspective

To develop the vectorization technique from the texture perspective, consider the simple eigenimage, or "eigenface", model for the space of grey level face images. The eigenface approach for modeling face images has been used recently for a variety of facial analysis tasks, including face recognition (Turk and Pentland [129], Akamatsu, *et al.* [5], Pentland, *et al.* [103]), reconstruction (Kirby and Sirovich [77]), face detection (Sung and Poggio [125], Moghaddam and Pentland [96]), and facial feature detection (Pentland, *et al.* [103]). The main assumption behind this modeling approach is that the space of grey level images of faces is linearly spanned by a set of example face images. To optimally represent this "face space", principal component analysis is applied to the example set, extracting an orthogonal set of eigenimages that define the dimensions of face space. Arbitrary faces are then represented by the set of coefficients computed by projecting the face onto the set of eigenimages.

One requirement on face images, both for the example set fed to principal components and for new images projected onto face space, is that they be geometrically normalized so that facial features line up across all images. Most normalization methods use a global transform, usually a similarity or affine transform, to align two or three major facial features. For example, in Pentland, *et al.* [103], the imaging apparatus effectively registers eyes, and Akamatsu, *et al.* [5] register the eyes and mouth.

However, because of the inherent variability of facial geometries across different people, aligning just a couple of features – such as the eyes – leaves other features misaligned. To the extent that some features are misaligned, even this normalized representation will confound differences in grey level information with differences in local facial geometry. This may limit the representation's generalization ability to new faces outside the original example set used for principal components. For example, a new face may match the texture of one particular linear combination of eigenimages but the shape may require another linear combination.

To decouple texture and shape, Craw and Cameron [45] and Shackelton and Welsh [118] represent shape separately and use it to geometrically normalize face texture by deforming it to a standard shape. Shape is defined by the (x, y) locations of a set of feature points, as in our definition for shape. In Craw and Cameron [45], 76 points outlining the eyes, nose, mouth, eyebrows, and head are used. To geometrically normalize texture using shape, image texture is deformed to a standard face shape, making it "shape free". This is done by first triangulating the image using the features and then texture mapping.

However, they did not demonstrate an effective automatic method for computing the vectorized shape/texture representation. This is mainly due to difficulties in finding correspondences for shape, where probably on the order of tens of features need to be located. Craw and Cameron [45] manually locate their features. Shackelton and Welsh [118], who focus on eye images, use the deformable template approach of Yuille, Cohen, and Hallinan [144] to locate eye features. However, for 19/60 of their example eye images, feature localization is either rated as "poor" or "no fit".

Note that in both of these approaches, computation of the shape and texture components have been separated, with shape being computed first. This differs from our approach, where shape and texture computations are interleaved in an iterative fashion. In their approach the link from shape to texture is present – using shape to geometrically normalize the input. But using a texture model to assist finding correspondences is not exploited.

6.2.3 Combining shape and texture

Our face vectorizer consists of two primary steps, a shape step that computes vectorized shape $\mathbf{y}_{a-std}^{std}$ and a texture step that uses the texture model to approximate the texture vector $\mathbf{t}_a$. Key to our vectorization procedure is linking the two steps in a mutually beneficial manner and iterating back and forth between the two until the representation converges. First, consider how the result of the texture step can be used to assist the shape step. Assuming for the moment that the texture step can provide an estimate $\widehat{\mathbf{t}}_a$ using equation (6.1), then the shape step estimates $\mathbf{y}_{a-std}^{std}$ by computing optical flow between the input and $\widehat{\mathbf{t}}_a$.

Next, to complete the loop between shape and texture, consider how the shape $\mathbf{y}_{a-std}^{std}$ can be used to compute the texture approximation $\widehat{\mathbf{t}}_a$. The shape $\mathbf{y}_{a-std}^{std}$ is used to geometrically normalize the input image using the backward warp

$$\mathbf{t}_a(\mathbf{x}) = i_a(\mathbf{x} + \mathbf{y}_{a-std}^{std}(\mathbf{x})),$$

where $\mathbf{x} = (x, y)$ is a 2D pixel location in standard shape. This normalization step aligns the facial features in the input image with those in the textures $\mathbf{t}_{p_j}$. Thus, when $\mathbf{t}_a$ is approximated in the texture step by projecting it onto the linear space spanned by the $\mathbf{t}_{p_j}$, facial features are properly registered.

Given initial conditions for shape and texture, our proposed system switches back and forth between texture and shape computations until a stable solution is found. Because of the manner in which the shape and texture computations feed back on each other, improving one component improves the other: better correspondences mean better feature alignment for textural analysis, and computing a better textural approximation improves the reference image used for finding correspondences. Empirically, we have found that the representation converges after several iterations.

Now that we have seen a general outline of our vectorizer, let us explore the details.

6.3 Basic Vectorization Method

The basic method for our vectorizer breaks down into two main parts, the off-line preparation of the example textures $\mathbf{t}_{p_j}$, and the on-line vectorization procedure applied to a new input image.

6.3.1 Off-line preparation of examples

The basic assumption made in modeling vectorized texture is that the space of face textures is linearly spanned by a set of geometrically normalized example face textures. Thus, in constructing a vectorizer we must first collect a group of representative faces that will define face space. Before using the example faces in the vectorizer, they are geometrically normalized to align facial features, and the grey levels are processed using principal components or the pseudoinverse to optimize run-time textural processing.

Geometric normalization

To geometrically normalize an example face, we apply a local deformation to the image to warp the face shape into a standard geometry. This local deformation requires both the shape of the example face as well as some definition of the standard shape. Thus, our off-line normalization procedure needs the face shape component for our example faces, something we provide manually. These manual correspondences are averaged to define the standard shape. Finally, a 2D warping operation is applied to do the normalization. We now go over these steps in more detail.

First, to define the shape of the example faces, a set of line segment features are positioned manually for each. The features, shown in Fig. 6-1, follow Beier and Neely's [13] manual correspondence technique for morphing face images (also see section 5.2.1). Pairing up image feature points into line segments gives one a natural control over local scale and rotation in the eventual deformation to standard shape, as we will explain later when discussing the deformation technique.

Next, we average the line segments over the example images to define the standard face shape (see Fig. 6-2). We don't have to use averaging – since we are creating a definition, we could have just chosen a particular example face. However, averaging

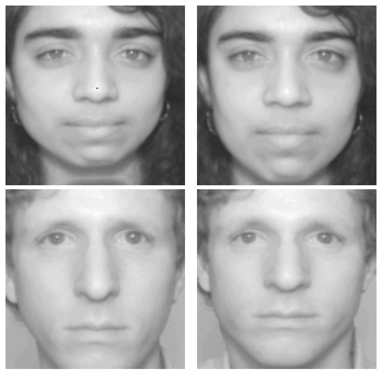


Figure 6-5: Examples of off-line geometrical normalization of example images. Texture for the normalized images is sampled from the original images – that is why the chin is generated for the second example.

shape should minimize the total amount of distortion required in the next step of geometrical normalization.

Finally, images are geometrically normalized using the local deformation technique of Beier and Neely [13]. This deformation technique is driven by the pairing of line segments in the example image with line segments in the standard shape. Consider a single pairing of line segments, one segment from the example image l_{ex} and one from the standard shape l_{std}. This line segment pair essentially sets up a local transform from the region surrounding l_{ex} to the region surrounding l_{std}. The local transform resembles a similarity transform except that there is no scaling perpendicular to the segment, just scaling along it. The local transforms are computed for each segment pair, and the overall warping is taken as weighted average. Some examples of images before and after normalization are shown in Fig. 6-5.

Texture processing

Now that the example faces have been normalized for shape, they can be used for texture modeling. Given a new input i_a, the texture analysis step tries to approximate the input texture $\mathbf{t}_a$ as a linear combination of the example textures. Of course, given a linear subspace such as our face space, one can choose among different sets of basis vectors that will span the same subspace. One popular method for choosing the basis set, the eigenimage approach, applies principal components analysis to the example set. Another potential basis set is simply the original set of images themselves. We now discuss the off-line texture processing required for the two basis sets of principal components and the original images.

Principal components analysis is a classical technique for reducing the dimensionality of a cluster of data points, where the data are assumed to be distributed in an ellipsoid pattern about a cluster center. If there is correlation in the data among the coordinate axes, then one can project the data points to a lower dimensional subspace without losing information. This corresponds to an ellipsoid with interesting variation along a number of directions that is less than the dimensionality of the data points. Principal components analysis finds the lower dimensional subspace inherent in the data points. It works by finding a set of directions $\mathbf{e}_i$ such that the variance in the data points is highest when projected onto those directions. These $\mathbf{e}_i$ directions are computed by finding the eigenvectors of the of the covariance matrix of the data points.

In our ellipsoid of n geometrically normalized textures $\mathbf{t}_{p_j}$, let $\mathbf{t}'_{p_j}$ be the set of textures with the mean $\mathbf{t}_{mean}$ subtracted off

$$\begin{aligned} \mathbf{t}_{mean} &= \frac{1}{n}\sum_{j=1}^{n} \mathbf{t}_{p_j} \\ \mathbf{t}'_{p_j} &= \mathbf{t}_{p_j} - \mathbf{t}_{mean}, \quad 1 \le j \le n. \end{aligned}$$

If we let T be a matrix where the jth column is $\mathbf{t}'_{p_j}$

$$T = \left[\mathbf{t}'_{p_1}\ \mathbf{t}'_{p_2}\ \cdots\ \mathbf{t}'_{p_n}\right],$$

then the covariance matrix is defined as

$$\Sigma = TT^t.$$

Notice that T is a $m \times n$ matrix, where m is the number of pixels in vectorized texture vectors. Due to our pixelwise representation for shape, $m \gg n$ and thus Σ, which is a $m \times m$ matrix, is quite large and may be intractable for eigenanalysis. Fortunately, one can solve the smaller eigenvector problem for the $n \times n$ matrix T^tT. This is possible because an eigenvector $\mathbf{e}_i$ of T^tT

$$T^tT\ \mathbf{e}_i = \lambda_i \mathbf{e}_i$$

corresponds to an eigenvector $T\mathbf{e}_i$ of Σ. This can be seen by multiplying both sides of the above equation by matrix T

$$(TT^t)\ T\mathbf{e}_i = \lambda_i T\mathbf{e}_i.$$

Since the eigenvectors (or eigenimages) $\mathbf{e}_i$ with the larger eigenvalues λ_i explain the most variance in the example set, only a fraction of the eigenimages need to be

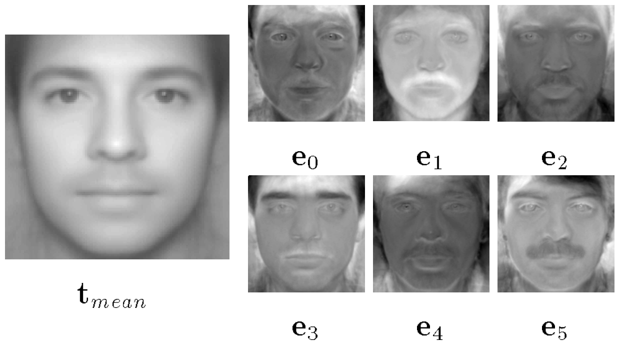


Figure 6-6: Mean image and eigenimages from applying principal components analysis to the geometrically normalized examples.

retained for the basis set. In our implementation, we chose to use roughly half the eigenimages. Fig. 6-6 shows the mean face and the first 6 eigenimages from a principal components analysis applied to a group of 55 people.

Since the eigenimages are orthogonal (and can easily be normalized to be made orthonormal), analysis and reconstruction of new image textures during vectorization can be easily performed. Say that we retain N eigenimages, and let $\mathbf{t}_a$ be a geometrically normalized texture to analyze. Then the run-time vectorization procedure projects $\mathbf{t}_a$ onto the $\mathbf{e}_i$

$$\beta_i = \mathbf{e}_i \cdot (\mathbf{t}_a - \mathbf{t}_{mean}) \tag{6.2}$$

and can reconstruct $\mathbf{t}_a$, yielding $\widehat{\mathbf{t}}_a$

$$\widehat{\mathbf{t}}_a = \mathbf{t}_{mean} + \textstyle\sum_{i=1}^{N} \beta_i \mathbf{e}_i. \tag{6.3}$$

Another potential basis set is the original example textures themselves. That is, we approximate $\mathbf{t}_a$ by a linear combination of the n original image textures $\mathbf{t}_{p_i}$

$$\widehat{\mathbf{t}}_a = \textstyle\sum_{i=1}^{n} \beta_i \mathbf{t}_{p_i}. \tag{6.4}$$

While we do not need to solve this equation until on-line vectorization, previewing the solution will elucidate what needs to be done for off-line processing. Write equation (6.4) in matrix form

$$\widehat{\mathbf{t}}_a = T\,\beta, \tag{6.5}$$

where $\widehat{\mathbf{t}}_a$ is written as a column vector, T is a matrix where the ith column is $\mathbf{t}_{p_i}$, and β is a column vector of the β_i's. Solving this with linear least squares yields

$$\begin{aligned} \beta &= T^{\dagger}\,\mathbf{t}_a && (6.6) \\ &= (T^t T)^{-1} T^t\,\mathbf{t}_a && (6.7) \end{aligned}$$

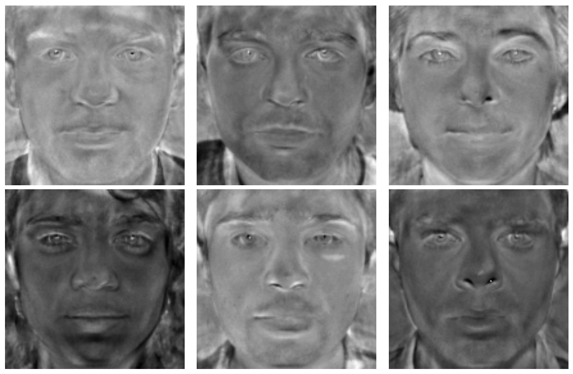


Figure 6-7: Example textures processed by the pseudoinverse $T^\dagger = (T^tT)^{-1}T^t$. When using the original set of image textures as a basis, texture analysis is performed by projection onto these images.

where $T^\dagger = (T^tT)^{-1}T^t$ is the pseudoinverse of T. The pseudoinverse can be computed off-line since it depends only on the example textures $\mathbf{t}_{p_i}$. Thus, run-time vectorization performs texture analysis with the columns of $T^\dagger$ (equation (6.6)) and reconstruction with the columns of T (equation (6.5)). Fig. 6-7 shows some example images processed by the pseudoinverse where n was 40.

Note that for both basis sets, the linear coefficients are computed using a simple projection operation. Coding-wise at run-time, the only difference is whether one subtracts off the mean image $\mathbf{t}_{mean}$. In practice though, the eigenimage approach will require fewer projections since not all eigenimages are retained. Also, the orthogonality of the eigenimages may produce a more stable set of linear coefficients – consider what happens for the pseudoinverse approach when two example images are similar in texture. Yet another potential basis set, one that has the advantage of orthogonality, would be the result of applying Gram-Schmidt orthonormalization to the example set.

Most of our vectorization experiments have been with the eigenimage basis, so the notation in the next section uses this basis set.

6.3.2 Run-time vectorization

In this section we go over the details of the vectorization procedure. The inputs to the vectorizer are an image i_a to vectorize and a texture model consisting of N eigenimages $\mathbf{e}_i$ and mean image $\mathbf{t}_{mean}$. In addition, the vectorizer takes as input a planar transform P that selects the face region from the image i_a and normalizes it for

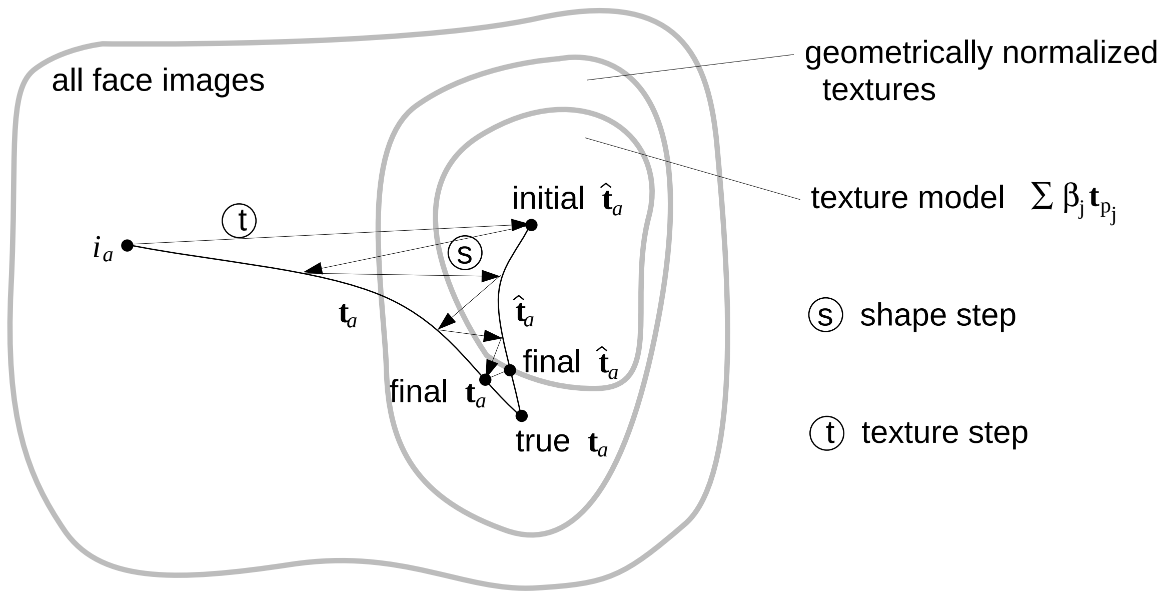

all face images
geometrically normalized textures
texture model $\Sigma \beta_j \mathbf{t}_{p_j}$
initial $\hat{\mathbf{t}}_a$
i_a
t
s
$\mathbf{t}_a$
$\hat{\mathbf{t}}_a$
final $\hat{\mathbf{t}}_a$
final $\mathbf{t}_a$
true $\mathbf{t}_a$
s shape step
t texture step

Figure 6-8: Convergence of the vectorization procedure with regards to texture. The texture and shape steps try to make $\hat{\mathbf{t}}_a$ and $\mathbf{t}_a$ converge to the true $\mathbf{t}_a$.

the effects of scale and image-plane rotation. The planar transform P can be a rough estimate from a coarse scale analysis. Since the faces in our test images were taken against a solid background, face detection is relatively easy and can be handled simply by correlating with a couple face templates. The vectorization procedure refines the estimate P, so the final outputs of the procedure are the vectorized shape $\mathbf{y}_{a-std}^{std}$, a set of β_i coefficients for computing $\hat{\mathbf{t}}_a$, and a refined estimate of P.

As mentioned previously, the interconnectedness of the shape and texture steps makes the iteration converge. Fig. 6-8 depicts the convergence of the vectorization procedure from the perspective of texture. There are three sets of face images in the figure, sets of (1) all face images, (2) geometrically normalized face textures, and (3) the space of our texture model. The difference between the texture model space and the set of geometrically normalized faces depends on the prototype set of n example faces. The larger and more varied this set becomes, the smaller the difference becomes between sets (2) and (3). Here we assume that the texture model is not perfect, so the true $\mathbf{t}_a$ is slightly outside the texture model space.

The goal of the iteration is to make estimates of $\mathbf{t}_a$ and $\hat{\mathbf{t}}_a$ converge to the true $\mathbf{t}_a$. The path for $\mathbf{t}_a$, the geometrically normalized version of i_a, is shown by the curve from i_a to the final $\mathbf{t}_a$. The path for $\hat{\mathbf{t}}_a$ is shown by the curve from initial $\hat{\mathbf{t}}_a$ to final

$\widehat{\mathbf{t}}_a$. The texture and shape steps are depicted by the arrows jumping between the curves. The texture step, using the latest estimate of shape to produce $\mathbf{t}_a$, projects $\mathbf{t}_a$ into the texture model space. The shape step uses the latest $\widehat{\mathbf{t}}_a$ to find a new set of correspondences, thus updating shape and hence $\mathbf{t}_a$. As one moves along the $\mathbf{t}_a$ curve, one is getting better estimates of shape. As one moves along the $\widehat{\mathbf{t}}_a$ curve, the β_i coefficients in the texture model improve. Since the true $\mathbf{t}_a$ lies outside the texture model space, the iteration stops at final $\widehat{\mathbf{t}}_a$. This error can be made smaller by increasing the number of prototypes for the texture model.

We now look at one iteration step in detail.

One iteration

In examining one iteration of the texture and shape steps, we assume that the previous iteration has provided an estimate for $\mathbf{y}^{std}_{a-std}$ and the β_i coefficients. For the first iteration, an initial condition of $\mathbf{y}^{std}_{a-std} = \vec{\mathbf{0}}$ is used. No initial condition is needed for texture since the iteration starts with the texture step.

In the **texture step**, first the input image i_a is geometrically normalized using the shape estimate $\mathbf{y}^{std}_{a-std}$, producing $\mathbf{t}_a$

$$\mathbf{t}_a(\mathbf{x}) = i_a(\mathbf{x} + \mathbf{y}^{std}_{a-std}(\mathbf{x})), \tag{6.8}$$

where $\mathbf{x} = (x, y)$ is a pixel location in the standard shape. This is implemented as a backwards warp using the flow vectors pointing from the standard shape to the input. As discussed in section 5.3.1, bilinear interpolation is used to sample i_a at non-integral (x, y) locations. Next $\mathbf{t}_a$ is projected onto the eigenimages $\mathbf{e}_i$ using equation (6.2) to update the linear coefficients β_i. These updated coefficients should enable the shape computation to synthesize an approximation $\widehat{\mathbf{t}}_a$ that is closer to the true $\mathbf{t}_a$.

In the **shape step**, first a reference image $\widehat{\mathbf{t}}_a$ is synthesized from the texture coefficients using equation (6.3). Since the reference image reconstructs the texture of the input, it should be well suited for finding shape correspondences. Next, optical flow is computed between $\widehat{\mathbf{t}}_a$, which is geometrically normalized, and i_a, which updates the pixelwise correspondences $\mathbf{y}^{std}_{a-std}$. For optical flow, we used the gradient-based hierarchical scheme of Bergen and Adelson [16], Bergen and Hingorani [18], and Bergen, *et al.* [17]. The new correspondences should provide better geometrical normalization in the next texture step.

Overall, iterating these steps until the representation stabilizes is equivalent to iteratively solving for the $\mathbf{y}^{std}_{a-std}$ and β_i which best satisfy

$$\mathbf{t}_a = \widehat{\mathbf{t}}_a,$$

or

$$i_a(\mathbf{x} + \mathbf{y}_{a-std}^{std}(\mathbf{x})) = \mathbf{t}_{mean} + \textstyle\sum_{i=1}^{n} \beta_i \mathbf{e}_i.$$

Adding a global transform

We introduce a planar transform P to select the image region containing the face and to normalize the face for the effects of scale and image-plane rotation. Let i'_a be the input image i_a resampled under the planar transform P

$$i'_a(\mathbf{x}) = i_a(P(\mathbf{x})). \tag{6.9}$$

It is this resampled image i'_a that will be geometrically normalized in the texture step and used for optical flow in the shape step.

Besides selecting the face, the transform P will also be used for selecting subimages around individual features such as the eyes, nose, and mouth. As will be explained in the next section on our hierarchical implementation, the vectorization procedure is applied in a coarse-to-fine strategy on a pyramid structure. Full face templates are vectorized at the coarser scales and individual feature templates are vectorized at the finer scales.

Transform P will be a similarity transform

$$P(\mathbf{x}) = s \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \mathbf{x} + \begin{pmatrix} t_x \\ t_y \end{pmatrix},$$

where the scale s, image-plane rotation θ, and 2D translation (t_x, t_y) are determined in one of two ways, depending on the region being vectorized.

1. *Two point correspondences.* Define anchor points $\mathbf{q}_{std,1}$ and $\mathbf{q}_{std,2}$ in standard shape, which can be done manually in off-line processing. Let $\mathbf{q}_{a,1}$ and $\mathbf{q}_{a,2}$ be estimates of the anchor point locations in the image i_a, estimates which need to be performed on-line. The similarity transform parameters are then determined such that

$$P(\mathbf{q}_{std,1}) = \mathbf{q}_{a,1}, \qquad P(\mathbf{q}_{std,2}) = \mathbf{q}_{a,2}. \tag{6.10}$$

   This uses the full flexibility of the similarity transform and is used when the image region being vectorized contains two reliable feature points such as the eyes.

2. *Fixed s, θ, and one point correspondence.* In this case there is only one anchor point $\mathbf{q}_{std,1}$, and one solves for t_x and t_y such that

$$P(\mathbf{q}_{std,1}) = \mathbf{q}_{a,1}. \tag{6.11}$$

This is useful for vectorizing templates with less reliable features such as the nose and mouth. For these templates the eyes are vectorized first and used to fix the scale and rotation for the nose and mouth.

While the vectorizer assumes that a face finder has provided an initial estimate for P, we would like the vectorizer to be insensitive to a coarse or noisy estimate and to improve the estimate of P during vectorization. The similarity transform P can be updated during the iteration when our estimates change for the positions of the anchor points $\mathbf{q}_{a,i}$. This can be determined after the shape step computes a new estimate of the shape $\mathbf{y}^{std}_{a-std}$. We can tell that an anchor point estimate is off when there is nonzero flow at the anchor point

$$\|\mathbf{y}^{std}_{a-std}(\mathbf{q}_{std,i})\| > \text{threshold}.$$

The correspondences can be used to update the anchor point estimate

$$\mathbf{q}_{a,i} = P(\mathbf{q}_{std,i} + \mathbf{y}^{std}_{a-std}(\mathbf{q}_{std,i})).$$

Next, P can be updated using the new anchor point locations using equation (6.10) or (6.11) and i_a can be resampled again using equation (6.9) to produce a new i'_a.

Entire procedure

The basic vectorization procedure is now summarized. Lines 2(a) and (b) are the texture step, lines 2(c) and (d) are the shape step, and line 2(e) updates the similarity transform P.

procedure vectorize

1. initialization
   (a) Estimate P using a face detector. For example, a correlational face finder using averaged face templates can be used to estimate the translational component of P.
   (b) Resample i_a using the similarity transform P, producing i'_a (equation (6.9)).
   (c) $\mathbf{y}^{std}_{a-std} = \vec{\mathbf{0}}$.
2. iteration: solve for $\mathbf{y}^{std}_{a-std}$, β_i, and P by iterating the following steps until the β_i stop changing.

(a) Geometrically normalize i'_a using $\mathbf{y}^{std}_{a-std}$, producing $\mathbf{t}_a$

$$\mathbf{t}_a(\mathbf{x}) = i'_a(\mathbf{x} + \mathbf{y}^{std}_{a-std}(\mathbf{x})).$$

(b) Project $\mathbf{t}_a$ onto example set $\mathbf{e}_i$, computing the linear coefficients β_i

$$\beta_i = \mathbf{e}_i \cdot (\mathbf{t}_a - \mathbf{t}_{mean}), \quad 1 \le i \le n.$$

(c) Compute reference image $\widehat{\mathbf{t}}_a$ for correspondence by reconstructing the geometrically normalized input

$$\widehat{\mathbf{t}}_a = \mathbf{t}_{mean} + \textstyle\sum_{i=1}^{n} \beta_i \mathbf{e}_i.$$

(d) Compute the shape component using optical flow

$$\mathbf{y}^{std}_{a-std} = \text{optical-flow}(i'_a, \widehat{\mathbf{t}}_a).$$

(e) If the anchor points are misaligned, as indicated by optical flow, then:

  i. Update P with new anchor points.

  ii. Resample i_a using the similarity transform P, producing i'_a (eqn (6.9)).

  iii. $\mathbf{y}^{std}_{a-std} = \text{optical-flow}(i'_a, \widehat{\mathbf{t}}_a)$.

Fig. 6-9 shows snapshot images of i'_a, $\mathbf{t}_a$, and $\widehat{\mathbf{t}}_a$ during each iteration of an example vectorization. The iteration number is shown in the left column, and the starting input is shown in the upper left. We deliberately provided a poor initial alignment for the iteration to demonstrate the procedure's ability to estimate the similarity transform P. As the iteration proceeds, notice how (1) improvements in P lead to a better global alignment in i'_a, (2) the geometrically normalized image $\mathbf{t}_a$ improves, and (3) the image $\widehat{\mathbf{t}}_a$ becomes a more faithful reproduction of the input. The additional row for i'_a is given because when step 2(e) is executed in the last iteration, i'_a is updated.

6.3.3 Pose dependence from the example set

The example images we have used in the vectorizer so far have been from a frontal pose. What about other poses, poses involving rotations out of the image plane?

Because we are being careful about geometry and correspondence, the example views used to construct the vectorizer must be taken from the same out-of-plane image rotation. The resulting vectorizer will be tuned to that pose, and performance is expected to drop as an input view deviates from that pose. The only thing that

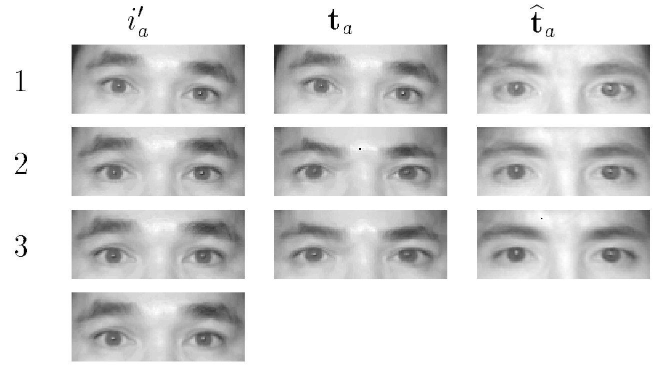


Figure 6-9: Snapshot images of i'_a, $\mathbf{t}_a$, and $\widehat{\mathbf{t}}_a$ during the three iterations of an example vectorization. See text for details.

makes the vectorizer pose-dependent, however, is the set of example views used to construct face space. The iteration step is general and should work for a variety of poses. Thus, even though we have chosen a frontal view as an example case, a vectorizer tuned for a different pose can be constructed simply by using example views from that pose.

In section 6.5 on applying the vectorizer to feature detection, we demonstrate two vectorizers, one tuned for a frontal pose, and one for an off-frontal pose. Later, in section 6.6.3, we suggest a multiple-pose vectorizer that connects different pose-specific vectorizers through interpolation.

6.4 Hierarchical implementation

For optimization purposes, the vectorization procedure is implemented using a coarse-to-fine strategy. Given an input image to vectorize, first the Gaussian pyramid (Burt and Adelson [33]) is computed to provide a multiresolution representation over 4 scales, the original image plus 3 reductions by 2. A face finder is then run over the coarsest level to provide an initial estimate for the similarity transform P. Next, the vectorizer is run at each pyramid level, working from the coarser to finer levels. As processing moves from a coarser level to a finer one, the coarse shape correspondences are used to initialize the similarity transform P for the vectorizer at the finer level.

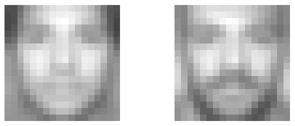


Figure 6-10: Face finding templates are grey level averages using two populations, all examples (left) plus people with beards (right).

6.4.1 Face finding at coarse resolution

For our test images, face detection is not a major problem since the subjects are shot against a uniform background. For the more general case of cluttered backgrounds, see the face detection work of Reisfeld and Yeshurun [115], Ben-Arie and Rao [14], Sung and Poggio [125], Sinha [122], and Moghaddam and Pentland [96]. For our test images, we found that normalized correlation using two face templates works well. The normalized correlation metric is

$$r = \frac{<\mathbf{TI}> - <\mathbf{T}><\mathbf{I}>}{\sigma(\mathbf{T})\sigma(\mathbf{I})},$$

where $\mathbf{T}$ is the template, $\mathbf{I}$ is the subportion of image being matched against, $<\mathbf{TI}>$ is normal correlation, $<>$ is the mean operator, and $\sigma()$ measures standard deviation. The templates are formed by averaging face grey levels over two populations, an average of all examples plus an average over people with beards. Before averaging, example face images are first warped to standard shape. Our two face templates for a frontal pose are shown in Fig. 6-10. To provide some invariance to scale, regions with high correlation response to these templates are examined with secondary correlations where the scale parameter is both increased and decreased by 20%. The location/scale of correlation matches above a certain threshold are reported to the vectorizer.

6.4.2 Multiple templates at high resolution

When processing the different pyramid levels, we use a whole face template at the two coarser resolutions and templates around the eyes, nose, and mouth for the two finer resolutions. This template decomposition across scales is similar to Burt's pattern tree approach [31] for template matching on a pyramid representation. At a coarse scale, faces are small, so full face templates are needed to provide enough spatial support for texture analysis. At a finer scale, however, individual features – eyes, noses – cover enough area to provide spatial support for analysis, giving us the option to perform separate vectorizations. The advantage of decoupling the analysis of the eyes, nose, and mouth is that it should improve generalization to new faces not in the original example set. For example, if the eyes of a new face use one set of linear texture coefficients and the nose uses another, separate vectorization for the eyes and nose provides the extra flexibility we need. However, if new inputs always come from people in the original example set, then this extra flexibility is not required and keeping to whole-face templates should be a helpful constraint.

When vectorizing separate eyes, nose, and mouth templates at the finer two resolutions, the template of the eyes has a special status for determining the scale and image-plane rotation of the face. The eyes template is vectorized first, using 2 iris features as anchor points for the similarity transform P. Thus, the eyes vectorization estimates a normalizing similarity transform for the face. The scale and rotation parameters are then fixed for the nose and mouth vectorizations. Only one anchor point is used for the nose and mouth, allowing only the translation in P to change.

6.4.3 Example results

For the example case in Fig. 6-11, correspondences from the shape component are plotted over the four levels of the Gaussian pyramid. These segment features are generated by mapping the averaged line segments from Fig. 6-2 to the input image. To get a sense of the final shape/texture representation computed at the highest resolution, Fig. 6-12 displays the final output for the Fig. 6-11 example. For the eyes, nose and mouth templates, we show i'_a, the geometrically normalized templates $\mathbf{t}_a$, and the reconstruction of those templates $\widehat{\mathbf{t}}_a$ using the linear texture coefficients. No images of this person were used among the examples used to create the eigenspaces.

We have implemented the hierarchical vectorizer in C on an SGI Indy R4600 based machine. Once the example images are loaded, multilevel processing takes just a few seconds to execute.

Experimental results presented in the next section on applications will provide a more thorough analysis of the vectorizer.

6.5 Application to feature finding

Once the vectorized representation has been computed, how can one use it? The linear texture coefficients can be used as a low-dimensional feature vector for face recognition, which is the familiar eigenimage approach to face recognition [129][5][103]. Our application of the vectorizer, however, has focused on using the correspondences in the shape component. In this section we describe experimental results from applying

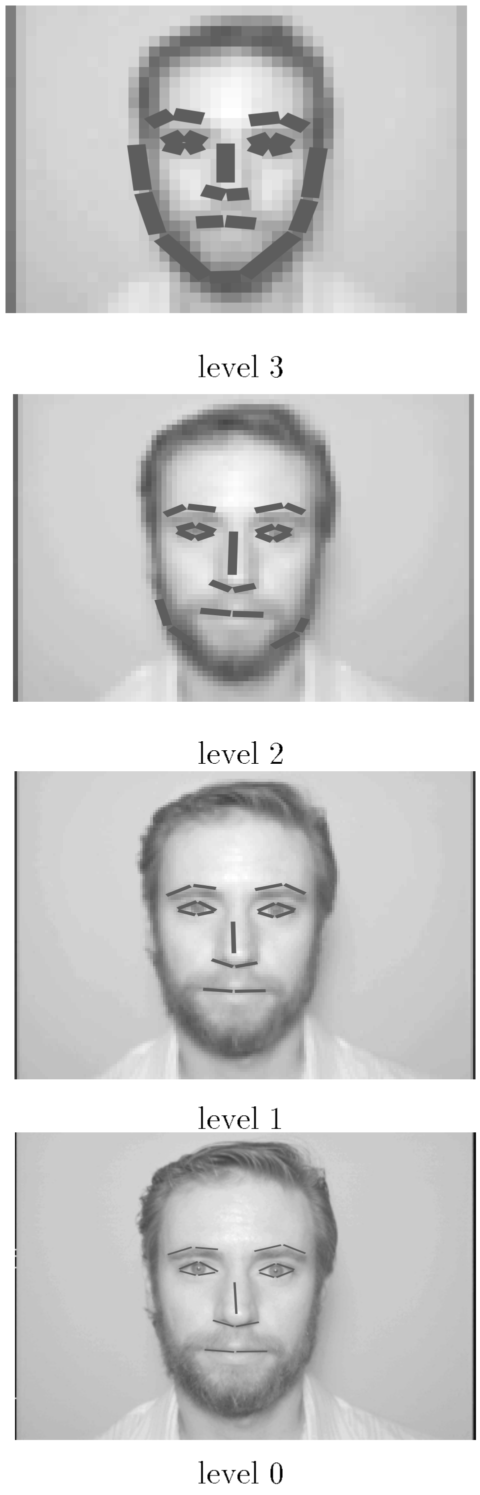


Figure 6-11: Evolution of the shape component during coarse-to-fine processing. The shape component is displayed through segment features which are generated by mapping the averaged line segments from Fig. 6-2 to the input image.

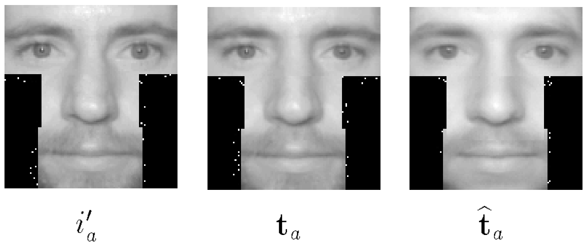


Figure 6-12: Final vectorization at the original image resolution.

these correspondences to the problem of locating facial features. Chapter 7 discusses how the shape correspondences can be used for synthesizing virtual views.

After vectorizing an input image i_a, pixelwise correspondence in the shape component $\mathbf{y}^{std}_{a-std}$ provides a dense mapping from the standard shape to the image i_a. Even though this dense mapping does more than locate just a sparse set of features, we can sample the mapping to locate a discrete set of feature points in i_a. To accomplish this, first, during off-line example preparation, the feature points of interest are located manually with respect to the standard shape. Then after the run-time vectorization of i_a, the feature points can be located in i_a by following the pixelwise correspondences and then mapping under the similarity transform P. For a feature point $\mathbf{q}_{std}$ in standard shape, its corresponding location in i_a is

$$P(\mathbf{q}_{std} + \mathbf{y}^{std}_{a-std}(\mathbf{q}_{std})).$$

For example, the line segment features of Fig. 6-2 can be mapped to the input by mapping each endpoint, as shown for the test images in Fig. 6-13.

In order to evaluate these segment features located by the vectorizer, two vectorizers, one tuned for a frontal pose (view **m3**, see Fig. 1-3) and one for a slightly rotated pose (view **m4**), were respectively tested on the **m3** and **m4** example views from our database. This image set consists of 62 people, 2 views per person – a frontal and slightly rotated pose – yielding a combined test set of 124 images. Example results from the rotated **m4** view vectorizer are shown in Fig. 6-14. Because the same views were used as example views to construct the vectorizers, a leave-6-out cross validation procedure was used to generate statistics. That is, the original group of 62 images from a given pose were divided into 11 randomly chosen groups (10 of 6 people, 1 of the remaining 2 people). Each group of images is tested using a different vectorizer; the vectorizer for group G is constructed from an example set consisting of the original images minus the set G. This allows us to separate the people used as examples from those in the test set.

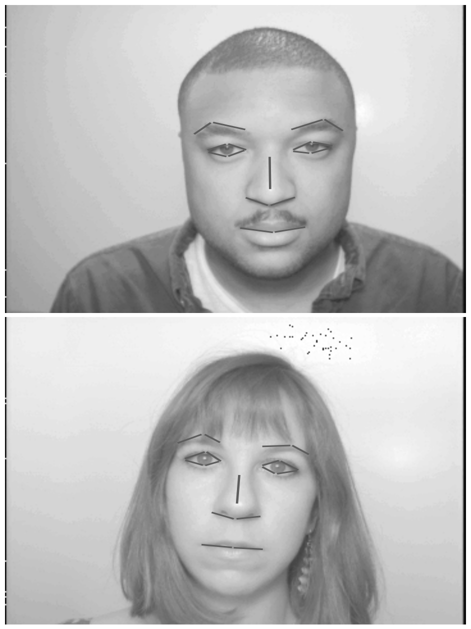


Figure 6-13: Example features located by sampling the dense set of shape correspondences $\mathbf{y}_{a-std}^{std}$ found by the vectorizer.

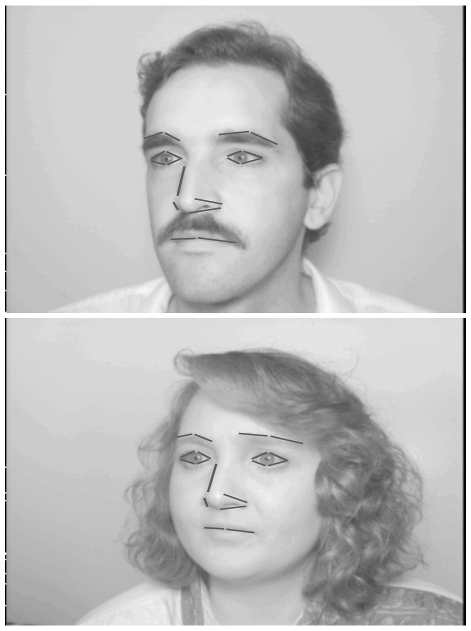


Figure 6-14: Example features located by the vectorizer.

feature	detection rate	average distances		
		point metric	edge metric	
		endpt. dist. (pixels)	angle (degrees)	perpend. dist. (pixels)
left eye	100% (124/124)	1.24	-	-
right eye	100% (124/124)	1.23	-	-
left eyebrow	97% (121/124)	-	5.1°	1.06
right eyebrow	96% (119/124)	-	4.8°	1.06
nose	99% (123/124)	1.45	3.2°	0.66
mouth	99% (123/124)	-	2.2°	0.53

Table 6.1: Detection rates and average distances between computed and "ground truth" segments. Qualitatively, the eyebrow and nose errors were misalignments, while the mouth error did involve a complete miss.

Qualitatively, the results were very good, with only one mouth feature being completely missed by the vectorizer (it was placed between the mouth and nose). To quantitatively evaluate the features, we compared the computed segment locations against manually located "ground truth" segments, the same segments used for off-line geometrical normalization. To report statistics by feature, the segments in Fig. 6-2 are grouped into 6 features: left eye (c_3, c_4, c_5, c_6), right eye (c_9, c_{10}, c_{11}, c_{12}), left eyebrow (c_1, c_2), right eyebrow (c_7, c_8), nose (n_1, n_2, n_3), and mouth (m_1, m_2).

Two different metrics were used to evaluate how close a computed segment came to its corresponding ground truth segment. Segments in the more richly textured areas (e.g. eye segments) have local grey level structure at both endpoints, so we expect both endpoints to be accurately placed. Thus, the "point" metric measures the two distances between corresponding segment endpoints. On the other hand, some segments are more edge-like, such as eyebrows and mouths. For the "edge" metric we measure the angle between segments and the perpendicular distance from the midpoint of the ground truth segment to the computed segment.

Next, the distances between the manual and computed segments were thresholded to evaluate the closeness of fit. A feature will be considered properly detected when *all* of its constituent segments are within threshold. Using a distance threshold of 10% of the interocular distance and an angle threshold of 20°, we compute detection rates and average distances between manual and computed segments (Table 6.1). The eyebrow and nose errors are more of a misalignment of a couple points rather than a complete miss (the mouth error was a complete miss).

6.6 Future work

In this section, first we discuss some shorter-term work for the existing vectorizer. This is followed by longer-term ideas for extending the vectorizer to use parameterized shape models and to handle multiple poses.

6.6.1 Existing vectorizer

So far the vectorizer has been tested on face images shot against a solid background. It would be nice to demonstrate the vectorizer working in cluttered environments. To accomplish this, both the face detection and vectorizer should be made more robust to the presense of false positive matches. To improve face detection, we would probably incorporate the learning approaches of Sung and Poggio [125] or Moghaddam and Pentland [96]. Both of these techniques model the space of grey level face images using principal components analysis. To judge the "faceness" of a image, they use a distance metric that includes two terms, "distance from face space" (see Turk and Pentland [129])

$$\|\mathbf{t}_a - \hat{\mathbf{t}}_a\|$$

and the Mahalanobis distance

$$\textstyle\sum_{i=1}^{N} \frac{\beta_i^2}{\lambda_i},$$

where the β_i are the eigenspace projection coefficients and λ_i are the eigenvalues from principal component analysis. This distance metric could be added to the vectorizer as a threshold test after the iteration step has converged.

Our current coarse-to-fine implementation does not exploit potential constraints that could be passed from the coarser to finer scales. The only information currently passed from a coarse level to the next finer level are feature locations used to initialize the similarity transform P. This could be expanded to help initialize the shape and texture components at the finer level as well.

6.6.2 Parameterized shape model

In the current vectorizer, shape is measured in a "data-driven" manner using optical flow. However, we can explicitly model shape by taking a linear combination of example shapes

$$\mathbf{y}_{a-std}^{std} = \textstyle\sum_{i=1}^{n} \alpha_i \mathbf{y}_{p_i-std}^{std},$$

where the shape of the ith example image, $\mathbf{y}_{p_i-std}^{std}$, is the 2D warping used to geometrically normalize the image in the off-line preparation step. This technique for modeling shape is similar to the work of Cootes, *et al.* [44], Blake and Isard [21], Baumberg and Hogg [12], and Jones and Poggio [72]. The new shape step would, given i'_a and reference $\widehat{\mathbf{t}}_a$, try to find a set of coefficients α_i that minimizes the squared error of the approximation

$$i'_a(\mathbf{x} + \textstyle\sum_{i=1}^{n} \alpha_i \mathbf{y}^{std}_{p_i-std}(\mathbf{x})) = \widehat{\mathbf{t}}_a.$$

This involves replacing the optical flow calculation with a model-based matching procedure; one can think of it as a parameterized "optical flow" calculation that computes a single set of linear coefficients instead of a flow vector at each point. One advantage of modeling shape is the extra constraint it provides, as some "illegal" warpings cannot even be represented. Additionally, compared to the raw flow, the linear shape coefficients should be more amenable for shape analysis tasks like expression analysis or face recognition using shape.

Given this new model for shape in the vectorizer, the set of α shape coefficients and β texture coefficients could be used as a low-dimensional representation for faces. An obvious application of this would be face recognition. Even without the modified vectorizer and the α coefficients, the β coefficients alone could be evaluated as a representation for a face recognizer.

6.6.3 Multiple poses

The straightforward way to handle different out-of-plane image rotations with the vectorizer is simply to use several vectorizers, each tuned to a different pose. However, if we provide pixelwise correspondence between the standard shapes of the different vectorizers, their operations can be linked together through image interpolation. The main idea is to interpolate among the $\widehat{\mathbf{t}}_a$ images of the different vectorizers to produce a new image that reconstructs both the grey levels *and the pose* of the input image (see Beymer, Shashua and Poggio [19] for examples of interpolation across different poses). Correspondence is then found between the input and this new interpolated image using optical flow. This correspondence, in turn, gives us correspondence between the input and the individual vectorizers, so the input can be warped to each one for a combined textural analysis. This procedure requires adding pose to the existing state variables of shape, texture, and similarity transform P. The output of this multi-pose vectorizer would be useful for pose estimation and pose-invariant face recognition.

6.7 Summary

This chapter explored an automatic technique for computing the vectorized representation that was introduced in the previous chapter. To design an algorithm for vectorizing images, or a "vectorizer", we observed that the shape and texture components of the representation can be linked. That is, for textural analysis, the shape component can be used to geometrically normalize an image so that features are properly aligned. Conversely, for shape analysis, the linear coefficients from the textural analysis can be used to create a reference image reconstructing the input. We then compute shape by finding correspondence between the reference image, which is at standard shape, and the input. The main idea of our vectorizer is to exploit the natural feedback between the texture and shape computations by iterating back and forth between the two until the shape/texture representation converges. We have demonstrated an efficient implementation of the vectorizer using a hierarchical coarse-to-fine strategy.

An application of the shape component to facial feature detection was explored. In our feature finding experiments, eyes, nose, mouth, and eyebrow features were located in 124 test images of 62 people at two different poses, and only one mouth feature was missed by the system. In the next chapter, we explore a second application of the shape component, the application of synthesizing virtual views.

Chapter 7

Face recognition using one view

As mentioned in the introduction, this thesis investigates two cases of the problem of pose-invariant face recognition: (i) multiple example views are available for each person, and (ii) only one example view per person is available. Chapter 4 discussed the first case, in which a simple view-based approach was taken to the problem. In this chapter we examine the second case, where, for example, perhaps just a driver's license photograph is available for each person in the database. If we wish to recognize new images of these people under a range of viewing directions, some of the new images will differ from the single view by a rotation in depth. Is recognition still possible?

7.1 Introduction

In general, there are a few potential approaches to the problem of face recognition from one example view. For example, the invariant features approach records features in the example view that do not change as pose-expression-lighting parameters change, features such as color or geometric invariants. While not yet applied to face recognition, this approach has been used for face detection under varying illumination (Sinha [122]) and for indexing of packaged grocery items using color (Swain and Ballard [126]).

In the flexible matching approach (von der Malsburg and collaborators [91][83], also see Chapter 2), the input image is deformed in 2D to match the example view. In [91], the deformation is driven by a matching of local "end-stop" features so that the resulting transformation between model and input is like a 2D warp rather than a global, rigid transform. This enables the deformation to match input and model views even though they may differ in expression or out-of-plane rotations. A deformation matching the input with a model view is evaluated by a cost functional that measures both the similarity of matched features and the geometrical distortion induced by the deformation. In this method, the difficulties include (a) constructing a generally valid cost functional, and (b) the computational expense of a non-convex optimization problem at run-time. However, since this matching mechanism is quite general (it does not take into consideration any prior model of human facial expression or 3D structure), it may be used for a variety of objects.

In the approach we explore in this thesis, prior knowledge of the object class (i.e. faces) is used to expand the limited example set by synthesizing virtual views. That is, given a single view of an object O and prior knowledge of the object class, additional views of O from other virtual viewpoints can be generated using the prior knowledge. As mentioned in sections 1.2 and 2.2, one example of representing this prior knowledge is a generic 3D model of the human face, which can be used to predict the appearance of a face under different pose-expression-lighting parameters. Once a 2D face image is texture mapped onto the 3D model, the face can be treated as a traditional 3D object in computer graphics, undergoing 3D rotations or changes in light source position.

In this thesis, we investigate representing prior knowledge of faces in an example-based manner, using 2D views of prototype faces. Since we address the problem of recognition under varying pose, the views of prototype faces will sample different rotations out of the image plane. In principle, though, different expressions and lightings can be modeled by sampling the prototype views under those parameters. Given one view of a person and the prototype views, we will propose in this chapter two methods for synthesizing virtual views of the person, linear classes and parallel deformation.

Our motivation for using the example-based approach is its potential for being a simple alternative to the more complicated 3D model-based approach. Using an example-based approach to bypass 3D models for 3D object recognition was first explored in the linear combinations approach to recognition (Ullman and Basri [130], Poggio [105]). In linear combinations, one can show that a 2D view of an object under rigid 3D transformation can be written as a linear combination of a small set of 2D example views, where the 2D view representation is the vectorized shape vector $\mathbf{y}$. This is valid for a range of viewpoints in which a number of feature points are visible in all views and thus can be brought into correspondence for the vectorized representation. This suggests an object may be represented using a set of 2D views instead of a 3D model.

Poggio and Vetter [110] have discussed this linear combinations approach in the case where only one example view is available for an object, laying the groundwork for virtual views. Normally, with just one view, 3D recognition is not possible. However, any method for generating additional object views would enable a recognition system to use the the linear combinations approach. This motivated Poggio and Vetter to introduce the idea of using prior knowledge of object class to generate virtual views.

Two types of prior knowledge were explored, knowledge of 3D object symmetry and example images of prototypical objects of the same class. In the former, the mirror reflection of the single example can be generated (also see section 1.2), and the latter leads to the idea of linear classes, which we will explain and use later in this chapter.

After discussing methods for generating virtual views, we evaluate their usefulness in the view-based, pose-invariant face recognizer from Chapter 4. Assuming that example view **m4** in Fig. 7-2 is the only real example view available for each person, we will synthesize the remaining set of 14 rotated virtual views. The combined set of one real and multiple virtual views will be used as example views in the view-based face recognizer. Recognition performance will be reported on the same set of testing images used in Chapter 4, which cover a range of rotations both in and out of the image plane.

Independent from our work, Lando and Edelman [84] have recently investigated the same overall question – generalization from a single view in face recognition – using a similar example-based technique for representing prior knowledge of faces. In addition, Maurer and von der Malsburg [93] have investigated a technique for transforming their "jet" features across rotations in depth. Their technique is more 3D than ours, as it uses a local planarity assumption and knowledge of local surface normals.

7.2 Prior knowledge of object class: prototype views

In our example-based approach for generating virtual views, prior knowledge of face transformations such as changes in rotation or expression are represented by 2D views of prototypical faces. Let there be N prototype faces $p_j, 1 \leq j \leq N$, where the prototypes are chosen to be representative of the variation in the class of faces. Unlike non-prototype faces – for which we only have a single example view – many views are available for each prototype p_j.

Given a single real view of a novel face at a known pose, we wish to transform the face to produce a rotated virtual view. Call the known pose of the real view the *standard* pose and the pose of the desired virtual view the *virtual* pose. Images of the prototype faces are then collected for both the standard and virtual poses. As shown in Fig. 7-1, let

$$\begin{aligned} i_{p_j} &= \text{set of } N \text{ prototype views at standard pose,} \\ i_{p_j,r} &= \text{set of } N \text{ prototype views at virtual pose,} \end{aligned}$$

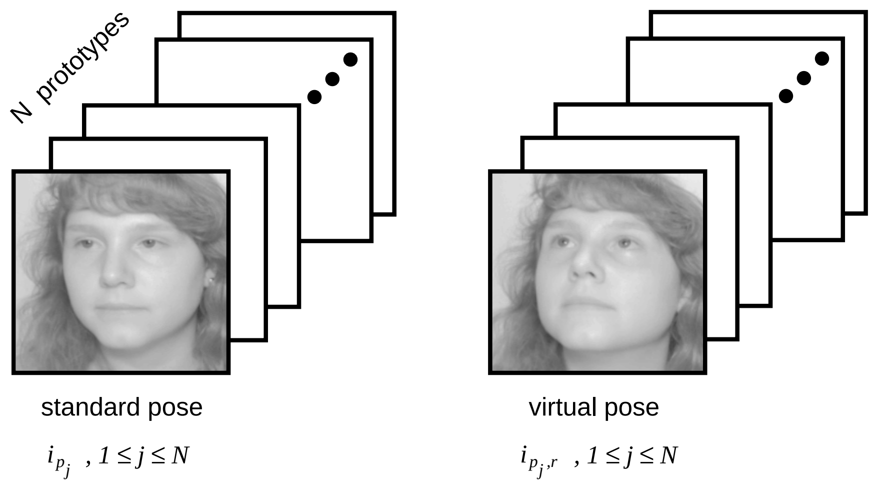


Figure 7-1: To represent prior knowledge of a facial transform (rotation upwards in the figure), views of N prototype faces are collected at the standard and virtual poses.

where $1 \le j \le N$. Since we wish to synthesize many virtual views from the same standard pose, sets of prototype views at the virtual pose will be acquired for all the desired virtual views.

The techniques we explore for generating virtual views work with the vectorized image representation introduced in Chapter 5. That is, the prototype images i_{p_j} and $i_{p_j,r}$ have been vectorized, producing shape vectors $\mathbf{y}_{p_j}$ and $\mathbf{y}_{p_j,r}$ and texture vectors $\mathbf{t}_{p_j}$ and $\mathbf{t}_{p_j,r}$. The specific techniques we used to vectorize images will be discussed in section 7.4.

In the vectorized image representation, a set of images are brought into correspondence by locating a common set of feature points across all images. Since the set of prototype views contain a variety of both people and viewpoints, our definition of the vectorized representation implies that correspondence needs to be computed across different viewpoints as well as different people. However, the two techniques for generating virtual views, parallel deformation and linear classes, have different requirements in terms of correspondence across viewpoint. Parallel deformation requires these correspondences, so the prototype views are vectorized as one large set. On the other hand, linear classes does not require correspondence across viewpoints, so the set of images is partitioned by viewpoint and separate vectorizations defined for each viewpoint. In this latter case, vectorization is simply handling correspondence across the different prototypes at a fixed pose.

7.3 Virtual views synthesis techniques

Let i_n be the single real view of the novel face in standard pose. In this section, we discuss how to synthesize the virtual view $i_{n,r}$ using the techniques of linear classes and parallel deformation.

7.3.1 Linear Classes

Because the theory of linear classes begins with a modeling assumption in 3D, let us generalize the 2D vectorized image representation to a 3D object representation. Recall that the 2D image vectorization is based on establishing feature correspondence across a set of 2D images. In 3D, this simply becomes finding a set of corresponding 3D points for a set of objects. The feature points are distributed over the face in 3D and thus may not all be visible from any one single view. Two measurements are made at each 3D feature point:

1. *Shape.* The (x, y, z) coordinates of the feature point. If there are n feature points, the vector $\mathbf{Y}$ will be a vector of length $3n$ consisting of the x, y, and z coordinate values.

2. *Texture.* If we assume that the 3D object is Lambertian and fix the lighting direction $\vec{l} = (l_x, l_y, l_z)$, we can measure the intensity of light reflected from each feature point, independent of viewpoint. At the ith feature point, the intensity $\mathbf{T}[i]$ is given by
$$\mathbf{T}[i] = \rho[i]\ (\vec{\eta}[i] \cdot \vec{l}), \tag{7.1}$$
where $\rho[i]$ is the albedo, or local surface reflectance, of feature i and $\vec{\eta}[i]$ is the local surface normal at feature i.[1]

The texture vector $\mathbf{T}$ is not an image; one can think of it as a texture that is mapped onto the 3D shape $\mathbf{Y}$ given a particular set of lighting conditions $\vec{l}$. One helpful way to visualize of the texture vector $\mathbf{T}$ is a sampling of image intensities in a cylindrical coordinate system that covers feature points over the entire face. This is similar to that produced by the Cyberware scanner.

Consider the relationship between 3D vectorized shape $\mathbf{Y}$ and texture $\mathbf{T}$ and their counterpart 2D versions $\mathbf{y}$ and $\mathbf{t}$. The projection process of going from 3D shape $\mathbf{Y}$ to

[1]Actually, it is not strictly necessary for the object to be Lambertian; equation (7.1) could be a different functional form of ρ, $\vec{\eta}$, and $\vec{l}$. What is necessary is that $\mathbf{T}[i]$ is independent of lighting and viewing direction, which may be achieved by fixing the light source and assuming that the object is Lambertian.

2D shape $\mathbf{y}$ consists of a 3D rotation, occlusion of a set of non-visible feature points, and orthographic projection. Mathematically, we model this using a matrix L

$$\mathbf{y} = L\mathbf{Y}, \tag{7.2}$$

where matrix L is the product of a 3D rotation matrix R, an occlusion matrix D that simply drops the coordinates of the occluded points, and orthographic projection O

$$L = ODR.$$

Note that L is a linear projection operator.

Creating a 2D texture vector $\mathbf{t}$ at a particular viewpoint $\vec{v}$ involves in some sense "projecting" the 3D texture $\mathbf{T}$. This is done by selecting the feature points that are visible in the standard shape at viewpoint $\vec{v}$

$$\mathbf{t} = D\mathbf{T}, \tag{7.3}$$

where D is a matrix that drops points occluded in the given viewpoint. Thus, viewpoint is handled in D; the lighting conditions are fixed in $\mathbf{T}$. Like operator L, D is a linear operator.

The idea of linear classes is based on the assumption that the space of 3D object vectorizations for objects of a given class is linearly spanned by a set of prototype vectorizations. That is, the shape $\mathbf{Y}$ and texture $\mathbf{T}$ of a class member can be written as

$$\mathbf{Y} = \sum_{j=1}^{N} \alpha_j \mathbf{Y}_{p_j} \quad \text{and} \quad \mathbf{T} = \sum_{j=1}^{N} \beta_j \mathbf{T}_{p_j} \tag{7.4}$$

for some set of α_j and β_j coefficients.

While the virtual views methods based on linear classes do not actually compute the 3D vectorized representation, the real view i_n is related to the destination virtual view $i_{n,r}$ through the 3D vectorization of the novel object. First, a 2D image analysis of i_n at standard pose estimates the α_j and β_j in equation (7.4) by using the prototype views i_{p_j}. Then the virtual view $i_{n,r}$ can be synthesized using the linear coefficients and the prototype views $i_{p_j,r}$. Let us now examine these steps in detail for the shape and texture of the novel face.

Virtual shape

Given the vectorized shape of the novel person $\mathbf{y}_n$ and the prototype vectorizations $\mathbf{y}_{p_j}$ and $\mathbf{y}_{p_j,r}, 1 \le j \le N$, linear classes can be used to synthesize vectorized shape $\mathbf{y}_{n,r}$ at the virtual pose. This idea was first developed by Poggio and Vetter [110].

In linear classes, we assume that the novel 3D shape $\mathbf{Y}_n$ can be written as a linear combination of the prototype shapes $\mathbf{Y}_{p_j}$

$$\mathbf{Y}_n = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{Y}_{p_j}. \tag{7.5}$$

If the linear class assumption holds and the set of 2D views $\mathbf{y}_{p_j}$ are linearly independent, then we can solve for the α_j's at the standard view

$$\mathbf{y}_n = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{y}_{p_j} \tag{7.6}$$

and use the prototype coefficients α_j to synthesize the virtual shape

$$\mathbf{y}_{n,r} = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{y}_{p_j,r}. \tag{7.7}$$

This is true under orthographic projection. The mathematical details are provided in Appendix B.

While this may seem to imply that we can perform a 3D analysis based on one 2D view of an object, the linear class assumption cannot be verified using 2D views. Thus, from just the 2D analysis, the technique can be "fooled" into thinking that it has found a good set of linear coefficients when in fact equation (7.5) is poorly approximated. That is, the technique will be fooled when the actual 3D shape of the novel person is different from the 3D interpolated prototype shape in the right hand side of equation (7.5).

In solving equations (7.6) and (7.7), the linear class approach can be interpreted as creating a direct mapping from standard to virtual pose. That is, we can derive a function that maps from $\mathbf{y}$'s in standard pose to $\mathbf{y}$'s in the virtual pose. Let Y be a matrix where column j is $\mathbf{y}_{p_j}$, and let Y_r be a matrix where column j is $\mathbf{y}_{p_j,r}$. Then if we solve for equation (7.6) using linear least squares and plug the resulting α's into equation (7.7), then

$$\mathbf{y}_{n,r} = Y_r Y^{\dagger} \mathbf{y}_n, \tag{7.8}$$

where $Y^{\dagger}$ is the pseudoinverse $(Y^t Y)^{-1} Y^t$.

Another way to formulate the solution as a direct mapping is to train a network to learn the association between standard and virtual pose (see Poggio and Vetter [110]). The (input, output) pairs presented to the network during training would be the prototype pairs $(\mathbf{y}_{p_j}, \mathbf{y}_{p_j,r})$. A potential architecture for such a network is suggested by the fact that equation (7.8) can be implemented by a single layer linear network. The weights between the input and output layers are given simply by the matrix $Y_r Y^{\dagger}$.

Virtual texture

In addition to generating the shape component of virtual views, the prototypes can also be used to generate the texture of virtual views. Given the texture of a novel face $\mathbf{t}_n$ and the prototype textures $\mathbf{t}_{p_j}$ and $\mathbf{t}_{p_j,r}, 1 \leq j \leq N$, the concept of linear classes can be used to synthesize the virtual texture $\mathbf{t}_{n,r}$. This synthesized grey level texture is then warped or texture mapped onto the virtual shape to create a finished virtual view. The ideas presented in this section were developed by the author and also independently by Vetter and Poggio [132].

To generate the virtual texture $\mathbf{t}_{n,r}$, we propose using the same linear class idea of approximation at the standard view and reconstruction at the virtual view. Similarly to the shape case, this relies on the assumption that the space of grey level textures $\mathbf{T}$ is linearly spanned by a set of prototype textures. The validity of this assumption is borne out by recent successful face recognition systems (e.g. eigenfaces, Pentland, *et al.* [103]). First, assume that the novel texture $\mathbf{T}_n$ can be written as a linear combination of the prototype textures $\mathbf{T}_{p_j}$

$$\mathbf{T}_n = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{T}_{p_j}. \tag{7.9}$$

The analog of linear classes for texture, presented in Appendix B, says that if this assumption holds and the 2D textures $\mathbf{t}_{p_j}$ are linearly independent, then we should be able to decompose the real texture $\mathbf{t}_n$ in terms of the example textures $\mathbf{t}_{p_j}$

$$\mathbf{t}_n = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j} \tag{7.10}$$

and use the same set of coefficients to reconstruct the texture of the virtual view

$$\mathbf{t}_{n,r} = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j,r}. \tag{7.11}$$

Note that the texture $\mathbf{T}$ and hence the β_j coefficients are dependent on the lighting conditions. Thus, by computing different views $\mathbf{t}$ using the D operator, we are effectively rotating the camera around the object. The geometry between object and light source is kept fixed.

We have synthesized textures for rotations of 10 to 15 degrees between standard and virtual poses with reasonable results; see section 7.4 for example $\mathbf{t}_{n,r}$ images and section 7.5 for recognition experiments. In terms of computing $\mathbf{t}_{n,r}$ from $\mathbf{t}_n$, we can use the same linear solution technique as for shape (equation (7.8)).

7.3.2 Parallel deformation

While the linear class idea does not require the $\mathbf{y}$ vectors to be in correspondence between the standard and virtual views, if we add such "cross view" correspondence then the linear class idea can be interpreted as finding a 2D deformation from $\mathbf{y}_n$ to $\mathbf{y}_{n,r}$. Having shape vectors in cross view correspondence simply means that the $\mathbf{y}$ vectors in both poses refer to the *same* set of facial feature points. The advantage of computing this 2D deformation is that the texture of the virtual view can be generated by texture mapping directly from the original view i_n. This avoids the need for additional techniques to synthesize virtual texture at the virtual view.

To see the deformation interpretation, subtract equation (7.6) from (7.7) and move $\mathbf{y}_n$ to the other side, yielding

$$\mathbf{y}_{n,r} = \mathbf{y}_n + \textstyle\sum_{j=1}^{N} \alpha_j (\mathbf{y}_{p_j,r} - \mathbf{y}_{p_j}). \tag{7.12}$$

Bringing shape vectors from the different poses together in the same equation is legal because we have added cross view correspondence. The quantity $\mathbf{\Delta y}_j = \mathbf{y}_{p_j,r} - \mathbf{y}_{p_j}$ is a 2D warp that specifies how prototype j's feature points move under the prototype transformation. Equation (7.12) modifies the shape $\mathbf{y}_n$ by a linear combination of these prototype deformations. The coefficients of this linear combination, the α_j's, are given by $Y^{\dagger}\mathbf{y}_n$, the solution to the approximation equation (7.6).

Consider as a special case the deformation approach with just one prototype. In this case, the novel face is deformed in a manner that imitates the deformation seen in the prototype. This is similar to performance-driven animation (Williams [136]), and Poggio and Brunelli [108], who call it *parallel deformation*, have suggested it as a computer graphics tool for animating objects when provided with just one view. Specializing equation (7.12) gives

$$\mathbf{y}_{n,r} = \mathbf{y}_n + (\mathbf{y}_{p,r} - \mathbf{y}_p), \tag{7.13}$$

where we have dropped the j subscripts on the prototype variable p. The deformation $\mathbf{\Delta y} = \mathbf{y}_{p,r} - \mathbf{y}_p$ essentially represents the prototype transform and is the same 2D warping as in the multiple prototypes case.

By looking at the one prototype case through specializing the original equations (7.6) and (7.7), we get $\mathbf{y}_n = \mathbf{y}_p$ and $\mathbf{y}_{n,r} = \mathbf{y}_{p,r}$. This seems to say that the virtual shape $\mathbf{y}_{n,r}$ is simply that of the prototype at virtual pose, so why should equation (7.13) give us anything different? However, the specialized equations, which approximate the novel shape by prototype shape, are likely to be poor approximations. Thus, we should really add error terms, writing $\mathbf{y}_n = \mathbf{y}_p + \mathbf{y}_{error_1}$ and $\mathbf{y}_{n,r} = \mathbf{y}_{p,r} + \mathbf{y}_{error_2}$. The error terms are likely to be highly correlated, so by subtracting the equations – as is done by parallel deformation – we cancel out the error terms to some degree.

7.3.3 Comparing linear classes and parallel deformation

What are some of the relative advantages of linear classes and parallel deformation? First, consider some of the advantages of linear classes over parallel deformation. Parallel deformation works well when the 3D shape of the prototype matches the 3D shape of the novel person. If the two 3D shapes differ enough, the virtual view generated by parallel deformation will appear geometrically distorted. Linear classes, on the other hand, effectively tries to construct a prototype that matches the novel shape by taking the proper linear combination of example prototypes. Another advantage of linear classes is that correspondence is not required between standard and virtual poses. Thus, linear classes may be able to cover a wider range of rotations out of the image plane as compared to parallel deformation.

One advantage of parallel deformation over linear classes is its ability to preserve peculiarities of texture such as moles or birthmarks. Parallel deformation will preserve such marks since it samples texture from the original real view of the novel person's face. For linear classes, it is most likely that a random mark on a person's face will be outside the linear texture space of the prototypes, so it will not be reconstructed in the virtual view.

7.4 Generating virtual views

In our approach to recognizing faces using just one example view per person, we first expand the example set by generating virtual views of each person's face. The full set of views that we would ultimately like to have for our view-based face recognizer are the set of 15 example views shown in Fig. 7-2 and originally used in the recognizer from Chapter 4. These views evenly sample the two rotation angles out of the image plane.

While Fig. 7-2 shows 15 real views, in virtual views we assume that only view **m4** is available and we synthesize the remaining 14 example views. For the single real view, an off-center view was favored over, say, a frontal view because of the recognition results for bilaterally symmetric objects of Poggio and Vetter [110]. When the single real view is from a nondegenerate pose (i.e. mirror reflection is not equal to original view), then the mirror reflection immediately provides a second view that can be used for recognition. The choice of an off-center view is also supported by the psychophysical experiments of Schyns and Bülthoff [116]. They found that when humans are trained on just one pose and tested on many, recognition performance is better when the single training view is an off-center one as opposed to a frontal pose.

In completing the set of 15 example views, the 8 views neighboring **m4** will be

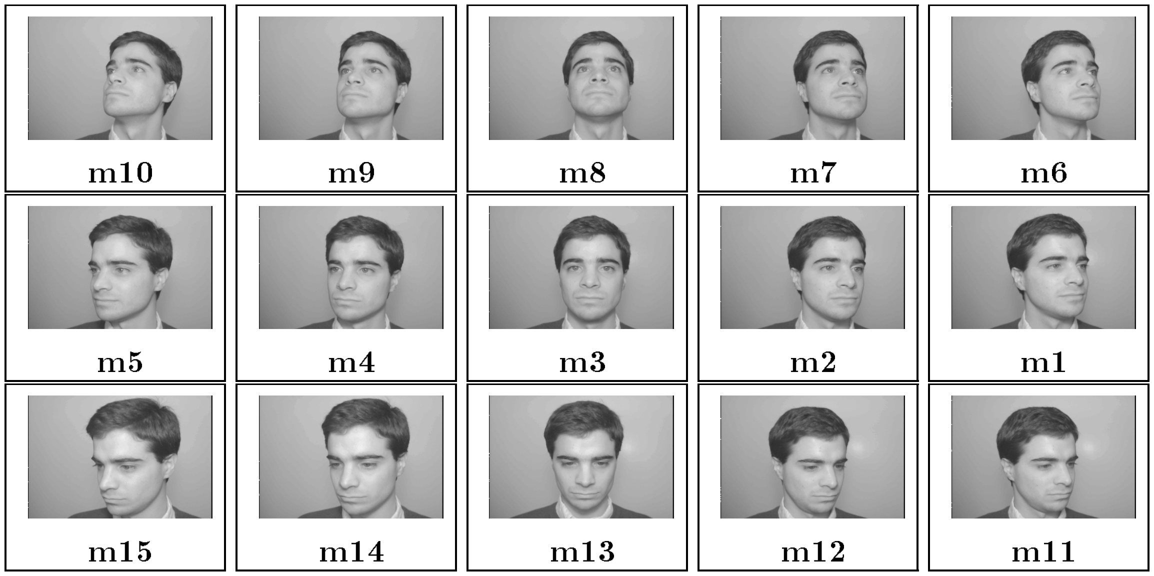


Figure 7-2: The view-based face recognizer uses 15 views to model a person's face. For virtual views, we assume that only one real view, view **m4**, is available and we synthesize the remaining 14.

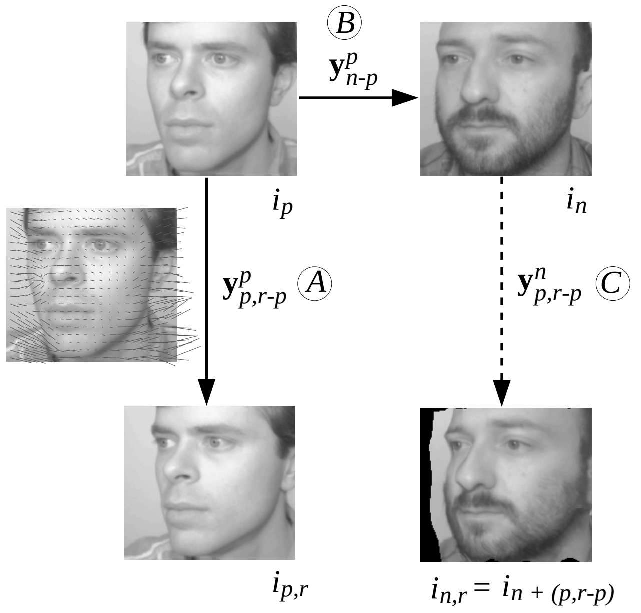
generated using our virtual views techniques. Using the terminology of the theory section, view **m4** is the standard pose and each of the neighboring views are virtual poses. The remaining 6 views, the right two columns of Fig. 7-2, will be generated by assuming bilateral symmetry of the face and taking the mirror reflection of the left two columns.

We now describe how parallel deformation and linear classes were used to expand the example set with virtual views. Recognition results with these virtual views are summarized in the next section.

7.4.1 Parallel deformation

The goal of parallel deformation is to map a facial transformation observed on a prototype face onto a novel, non-prototype face. There are three steps in implementing parallel deformation: (a) recording the deformation $\mathbf{y}_{p,r} - \mathbf{y}_p$ on the prototype face, (b) mapping this deformation onto the novel face, and (c) 2D warping the novel face using the deformation. We now go over these steps in more detail, using as an example the prototype views and single novel view in Fig. 7-3.

First, we collect prototype views i_p and $i_{p,r}$ and compute the prototype deforma-



B
$\mathbf{y}^p_{n\text{-}p}$
i_p
i_n
$\mathbf{y}^p_{p,r\text{-}p}$ A
$\mathbf{y}^n_{p,r\text{-}p}$ C
$i_{p,r}$
$i_{n,r} = i_{n\,+\,(p,r\text{-}p)}$

Figure 7-3: In parallel deformation, (A) the prototype flow $\mathbf{y}^p_{p,r-p}$ is first measured between $i_{p,r}$ and i_p, (B) the flow is mapped onto the novel face i_n, and (C) the novel face is 2D warped to the virtual view.

tion

$$\mathbf{y}^p_{p,r-p} = \mathbf{vect}(i_{p,r}, i_p)$$

using optical flow. Shown overlayed on the reference image on the left of Fig. 7-3, this 2D deformation specifies how to forward warp i_p to $i_{p,r}$ and represents our "prior knowledge" of face rotation. To assist the correspondence calculation, a sequence of four frames from standard to virtual pose is used instead of just two frames. Pairwise optical flows are computed and concatenated to get the composite flow from first to last frame.

Next, the 2D rotation deformation is mapped onto the novel person's face by changing the reference frame of $\mathbf{y}^p_{p,r-p}$ from i_p to i_n. First, interperson correspondences between i_p and i_n are computed

$$\mathbf{y}^p_{n-p} = \mathbf{vect}(i_n, i_p)$$

and used to change the reference frame

$$\mathbf{y}^n_{p,r-p} = \mathbf{fwarp\text{-}vect}(\mathbf{y}^p_{p,r-p}, \mathbf{y}^p_{n-p}).$$

The flow $\mathbf{y}^n_{p,r-p}$ is the 2D rotation deformation mapped onto the novel person's standard view. As the interperson correspondences are difficult to compute, we evaluated

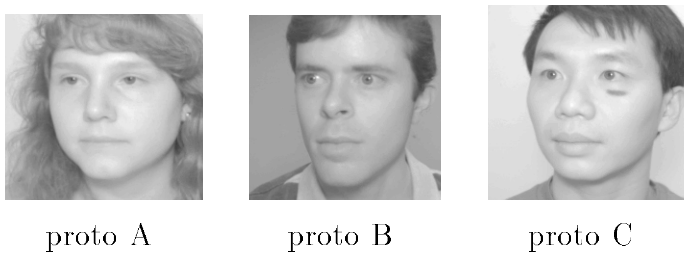


Figure 7-4: The prototypes used for parallel deformation. Standard poses are shown.

two techniques for establishing feature correspondence: labeling features manually on both faces, and using the vectorizer from Chapter 6 to automatically locate features. More will be said about these two approaches shortly.

Finally, the texture from the original real view i_n is 2D warped onto the rotated face shape, producing the final virtual view

$$i_{n,r} = i_{n+(p,r-p)} = \mathbf{fwarp}(i_n, \mathbf{y}^n_{p,r-p}).$$

Referring to our running example in Fig. 7-3, the final virtual view is shown in the lower right.

In this procedure for parallel deformation, there are two main parameters that one may vary:

1. *The prototype.* As mentioned previously, the accuracy of virtual views generated by parallel deformation depends on the degree to which the 3D shape of the prototype matches the 3D shape of the novel face. Thus, one would expect different recognition results from different prototypes. We have experimented with virtual views generated using the three different prototypes shown in Fig. 7-4. In general, given a particular novel person, it is best to have a variety of prototypes to choose from and to try to select the one that is closest to the novel person in terms of shape.

2. *Approach for interperson correspondence.* In both the manual and automatic approaches, interperson correspondences are driven by the line segment features shown in Fig. 7-5. The automatic segments shown on the right were located using our face vectorizer from Chapter 6. The manual segments on the left include some additional features not returned by the vectorizer, especially around the sides of the face. Given these sets of correspondences, the interpolation method from Beier and Neely [13] (see section 5.2.1) is used to interpolate the correspondences to define a dense, pixelwise mapping from the prototype to

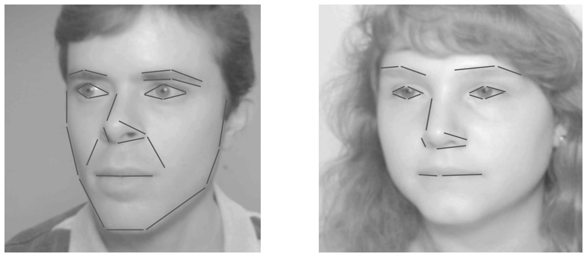


Figure 7-5: Parallel deformation requires correspondences between the prototype and novel person. These correspondences are driven by the segment features shown in the figure. The features on the left were manually located, and the features on the right were automatically located using the vectorizer.

novel face. For the automatic case, we did try to use the dense $\mathbf{y}$ shape vectors directly to avoid the Beier and Neely interpolant. However, the vectorizer limits correspondence to areas around the eyes, nose, mouth, which means that virtual views are defined only in those regions. This, in turn, made the optical flow correspondence step in the recognizer itself more difficult. The Beier and Neely interpolation method provides a simple way to extrapolate the correspondences defined over the center part of the face to the face periphery.

Figures 7-6 and 7-7 show example virtual views generated using prototype A with the real view in the center. Manual interperson correspondences were used in Fig. 7-6 and the image vectorizer in Fig. 7-7. To compare views generated from the different prototypes, Fig. 7-8 shows virtual views generated from all three prototypes. For comparison purposes, the real view of each novel person is shown on the right.

7.4.2 Linear Classes

We use the linear class idea to analyze the novel texture in terms of the prototypes at the standard view and reconstruct at the virtual view. In the analysis step at the standard view, we decompose the shape free texture of the novel view $\mathbf{t}_n$ in terms of the N shape free prototype views $\mathbf{t}_{p_j}$

$$\mathbf{t}_n = \sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j}, \tag{7.14}$$

which results in a set of β_j prototype coefficients. But before solving this equation for the β_j, the novel view i_n and prototype views i_{p_j} must be vectorized to produce

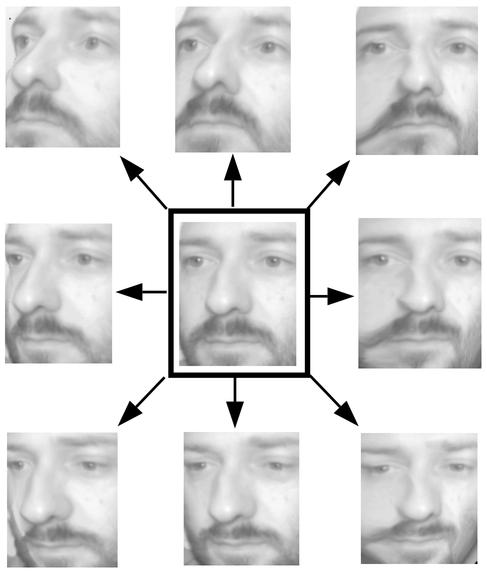


Figure 7-6: Example virtual views using parallel deformation. Prototype A was used, and interperson correspondence $\mathbf{y}_{n-p}^{p}$ was specified manually.

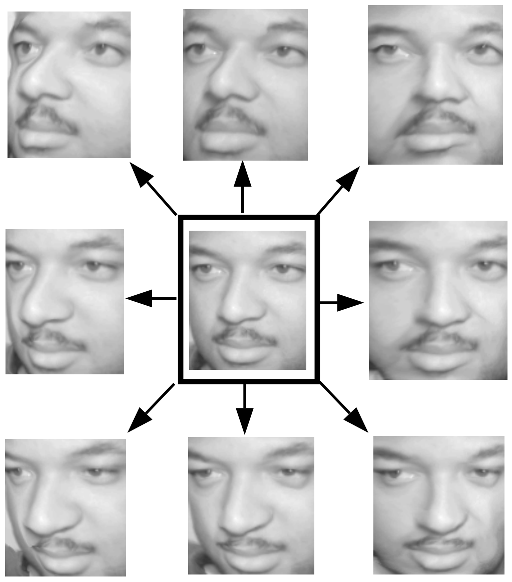


Figure 7-7: Example virtual views using parallel deformation. Prototype A was used, and interperson correspondence $\mathbf{y}_{n-p}^{p}$ was computed automatically using the image vectorizer.

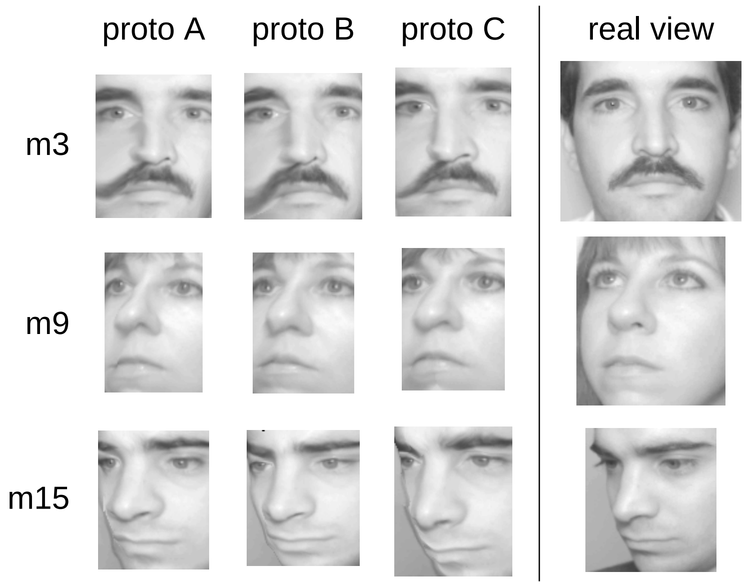


Figure 7-8: Example virtual views as the prototype person is varied. The corresponding real view of each novel person is shown on the right for comparison.

the geometrically normalized textures $\mathbf{t}_n$ and $\mathbf{t}_{p_j}, 1 \leq j \leq N$. Since the $\mathbf{t}_{p_j}$'s can be put into correspondence manually in an off-line step (using the Beier and Neely approach discussed in section 5.2.1), the primary difficulty of this step is in converting i_n into its shape free representation $\mathbf{t}_n$. Since i_n is an **m4** view of the face, this step means finding correspondence between i_n and view **m4**'s standard face shape. Let this standard shape be denoted as $\mathbf{y}_{std}$.

Our image vectorizer, which we describe in Chapter 6, is used to solve for the correspondences $\mathbf{y}^{std}_{n-std}$ between i_n and standard shape $\mathbf{y}_{std}$. These correspondences can then be used to geometrically standardize i_n

$$\mathbf{t}_n(\mathbf{x}) = i_n(\mathbf{x} + \mathbf{y}^{std}_{n-std}(\mathbf{x})),$$

where $\mathbf{x}$ is an arbitrary 2D point (x, y) in standard shape. Fig. 7-9 on the left shows an example view i_n with some features automatically located by the vectorizer. The right side of the figure shows templates $\mathbf{t}_n$ of the eyes, nose, and mouth that have been geometrically normalized using the correspondences $\mathbf{y}^{std}_{n-std}$.

Next, the texture $\mathbf{t}_n$ is decomposed as a linear combination of the prototype textures, following equation (7.14). First, combine the β_j terms into a column vector β and define a matrix T of the prototype textures, where the jth column of T is $\mathbf{t}_{p_j}$.

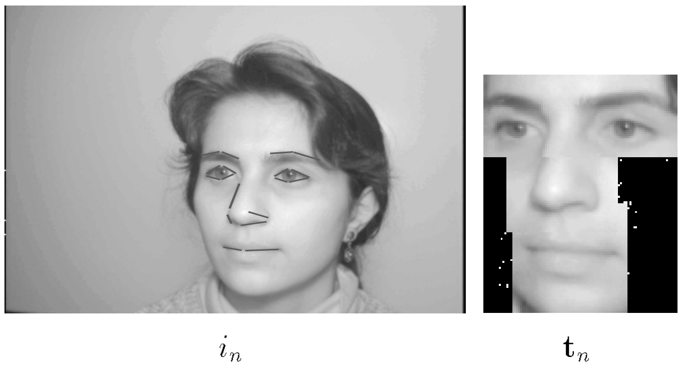


Figure 7-9: Using correspondences from our face vectorizer, we can geometrically normalize input i_n, producing the "shape free" texture $\mathbf{t}_n$.

Then equation (7.14) can be rewritten as

$$\mathbf{t}_n = T\beta.$$

This can be solved using linear least squares, yielding

$$\beta = T^{\dagger}\mathbf{t}_n,$$

where $T^{\dagger}$ is the pseudoinverse $(T^tT)^{-1}T^t$.

The synthesis step assumes that the textural decomposition at the virtual view is the same as that at the standard view. Thus, we can synthesize the virtual texture

$$\mathbf{t}_{n,r} = \sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j,r},$$

where $\mathbf{t}_{p_j,r}$ are the shape free prototypes that have been warped to the standard shape of the virtual view. As with the $\mathbf{t}_{p_j}$'s, the $\mathbf{t}_{p_j,r}$'s are put into correspondence manually in an off-line step. If we define a matrix T_r such that column j is $\mathbf{t}_{p_j,r}$, the analysis and synthesis steps can be written as a linear mapping from $\mathbf{t}_n$ to $\mathbf{t}_{n,r}$

$$\mathbf{t}_{n,r} = T_r T^{\dagger}\mathbf{t}_n.$$

This linear mapping was previously discussed in section 7.3.1 for generating virtual shapes.

Fig. 7-10 shows a set of virtual views generated using the analysis of Fig. 7-9. Note that the prototype views must be of the same set of people across all nine views. We used a prototype set of 55 people, so we had to specify manual correspondence (see

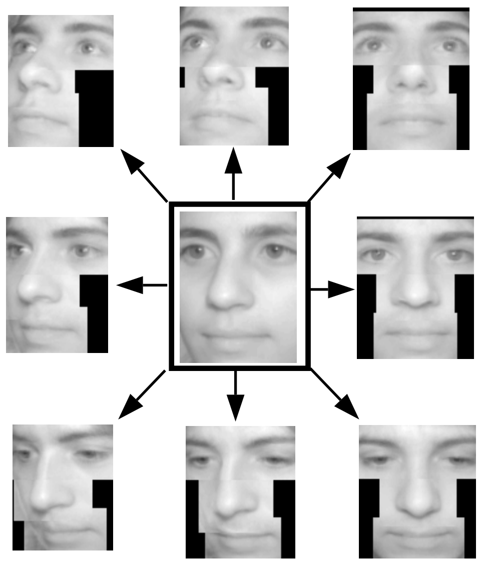


Figure 7-10: Example virtual views for linear classes.

Fig. 6-1) for 9 views of each person to set up the shape free views. When generating the virtual views for a particular person, we would, of course, remove him from the prototype set if he were initially present, following a cross validation methodology.

Notice from Fig. 7-10 that by using the shape free textural representation, the virtual views in this experiment are decoupled from shape and hence all views are in the standard shape of the virtual pose. The only difference between the views of different people at a fixed pose will be their texture.

7.5 Experimental results

In this section we report the recognition rates obtained when virtual views were used in our view-based recognizer from Chapter 4. The recognizer is essentially the same registration followed by correlation mechanism. For testing linear classes, there are some minor changes to the optical flow step since the whole face template is not available. In this case, optical flow is performed over individual templates rather than the entire face.

To test the recognizer, a set of 10 testing views per person were taken to randomly sample poses within the overall range of poses in Fig. 7-2. Roughly half of the test views include an image-plane rotation, so all three rotational degrees of freedom are tested. There are 62 people in the database, including 44 males and 18 females,

interperson correspondence	prototype		
	A	B	C
manual	84.5%	83.9%	83.9%
auto	85.2%	84.0%	83.4%

Table 7.1: Recognition rates for parallel deformation for the different prototypes and for manual vs. automatic features.

people from different races, and an age range from the 20s to the 40s. Lighting for all views is frontal and facial expression is neutral.

Table 7.1 shows recognition rates for parallel deformation for the different prototypes and for manual vs. automatic features. As with the experiments with real views in Chapter 4, the recognition rates were recorded for a forced choice scenario – the recognizer always reports the best match. In the template-based recognizer, template scale was fixed at an intermediate scale (interocular distance = 30 pixels) and preprocessing was fixed at dx+dy (the sum of separate correlations on the x and y components of the gradient). These parameters had yielded the best recognition rates for real views in Chapter 4. The results were fairly consistent, with a mean recognition rate of 84.1% and a standard deviation of only 0.6%. Automatic feature correspondence on average was as good as the manual correspondences, which was a good result for the face vectorizer. In the manual case, though, it is important to note that the manual step is at "model-building" time; the face recognizer at run time is still completely automatic.

Fig. 7-11 summarizes our experiments with using real and virtual views in the recognizer. Starting on the right, we repeat the result from Chapter 4 where we use 15 real views per person. This recognition rate of 98.7% presents a "best case" scenario for virtual views. The real views case is followed by parallel deformation, which gives a recognition rate of 85.2% for prototype A and automatic interperson correspondences. Next, linear classes on texture yields a recognition rate of 73.5%. To put these two recognition numbers in context, we compare them to a "base" case that uses only two example views per person, the real view **m4** plus its mirror reflection. A recognition rate of 70% was obtained for this two view case, thus establishing a lower bound for virtual views. Parallel deformation at 85% falls midway between the benchmark cases of 70% (one view + mirror reflect.) and 98%, (15 views) so it shows that virtual views do benefit pose-invariant face recognition.

In addition, the leftmost bar in Fig. 7-11 (one view) gives the recognition rate when only the view **m4** is used. This shows how much using mirror reflection helps

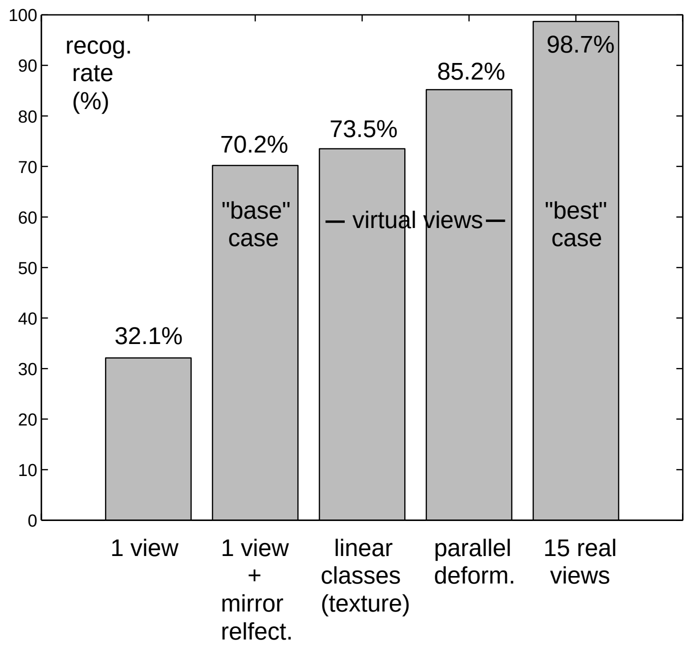


Figure 7-11: Face recognition performance for real and virtual views.

in the single real view case: without the view generated by mirror reflection, the recognition rate is roughly cut in half from 70% to 32%. This low recognition rate is caused by winnowing of example views based on the coarse pose estimate (looking left vs. looking right) of the input. If the input view is "looking right", then the system does not even try to match against the **m4** example view, which is "looking left". In this (one view) case, 62% of the inputs are rejected, and 6% of the inputs give rise to substitution errors.

Linear classes for virtual texture was a disappointment, however, only yielding a recognition rate a few percentage points higher than the base case of 70%. This may have been due to the factoring out of shape information. We also noticed that the linear reconstruction has a "smoothing" effect, reproducing the lower frequency components of the face better than the higher frequency ones. One difference in the experimental test conditions with respect to parallel deformation was that correlation was performed on the original grey levels instead of dx+dy; empirically we obtained much worse performance after applying a differential operator.

7.6 Discussion

7.6.1 Evaluation of recognition rate

While the recognition rate using virtual views, ranging from 85% for parallel deformation to 73% for linear classes, is much lower than the 98% rate for the multiple views case, this was expected since virtual views use much less information. One way to evaluate these rates is to use human performance as a benchmark. To test human performance, one would provide a subject with a set of training images of previously unknown people, using only one image per person. After studying the training images, the subject would be asked to identify new images of the people under a variety of poses. Moses, Ullman, and Edelman [97] have performed this experiment using testing views at a variety of poses and lighting conditions. While high recognition rates were observed in the subjects (97%), the subjects were only asked to discriminate between three different people. Bruce [25] performs a similar experiment where the subject is asked whether a face had appeared during training, and detection rates go down to either 76% or 60%, depending on the amount of pose/expression difference between the testing and training views. Schyns and Bülthoff [116] obtain a low recognition rate, but their results are difficult to compare since their stimuli are Gouraud shaded 3D faces that exclude texture information. Lando and Edelman [84] have recently performed computational experiments to replicate earlier psychophysical results in [97]. A recognition rate of only 76% was reported, but the authors suggest that this may be improved by using a two-stage classifier instead of a single-stage one.

Direct comparison of our results to related face recognition systems is difficult because of differences in example and testing views. The closest systems are those of Lando and Edelman [84] and Maurer and von der Malsburg [93]. Both systems explore a view transformation method that effectively generates new views from a single view. The view representation, in contrast to our template-based approach, is feature-based: Lando and Edelman use difference of Gaussian features, and Maurer and von der Malsburg use a set of Gabor filters at a variety of scales and rotations (called "jets"). The prior knowledge Lando and Edelman used to transform faces is similar to ours, views of prototype faces at standard and virtual views. They average the transformation in feature space over the prototypes and apply this average transformation to a novel object to produce a "virtual" set of features. As mentioned above, they report a recognition rate of 76%. Maurer and von der Malsburg transform their Gabor jet features by approximating the facial surface at each feature point as a plane and then estimating how the Gabor jet changes as the plane rotates in 3D. They apply this technique to rotating faces about 45° between frontal and half-profile views. They report a recognition rate of 53% on a subset of 90 people from the FERET database.

Two other comparable results are from Manjunath, *et al.* [91], who obtain 86% on a database of 86 people, and Pentland, *et al.* [103], whose extrapolation experiment with view-based eigenspaces yields 83% on a database of 21 people. In both cases, the system is trained on a set of views (vs. just one for ours) and recognition performance is tested on views from outside the pose-expression space of the training set. One difference in example views is that they include hair and we do not. In the future, the new Army FERET database should provide a common benchmark for comparing recognition algorithms.

7.6.2 Difficulties with virtual views generation

Since we know that the view-based approach performs well with real example views, making the virtual views closer in appearance to the "true" rotated views would obviously improve recognition performance. What difficulties do we encounter in generating "true" virtual views? First, the parallel deformation approach for shape essentially approximates the 3D shape of the novel person with the 3D shape of the prototype. If the two 3D shapes are different, the virtual view will not be "true" even though it may still appear to be a valid face. The resulting shape is a mixture of the novel and prototype shapes. Using multiple prototypes and the linear class approach may provide a better shape approximation.

In addition, for parallel deformation we have problems with areas that are visible in the virtual view but not in the standard view. For example, for the **m4** pose, the underside of the nose is often not visible. How can one predict how that region appears for upward looking virtual views? Possible ways to address this problem include using additional real views or having the recognizer exclude those regions during matching.

7.6.3 Transformations besides rotation

While the theory and recognition experiments in this paper revolve around generating rotated virtual views, one may also wish to generate virtual views for different lighting conditions or expressions. This would be useful for building a view-based face recognizer that handles those kinds of variation in the input. Here we suggest ways to generate these views.

Lighting

For changes in lighting conditions, the prototype faces are fixed in pose but the position of the light source is changed between the standard and virtual views. Unfortunately, changing the direction of the light source violates an assumption made for linear classes that the lighting conditions are fixed. That assumption had allowed us to ignore the fact that surface albedo and the local surface normal are confounded in the Lambertian model for image intensity.

However, the idea of parallel deformation can still be applied. Parallel deformation assumes that the 3D shape of the prototype is similar to the 3D shape of the novel person. Thus, corresponding points on the two faces should have the same local surface normal. The following analysis focuses on the image brightness of the same feature point on both the prototype and novel face. The two feature points may have been brought into correspondence through a vectorization procedure. Let

η	=	surface normal for both the prototype and novel faces
l_{std}	=	light source direction for standard lighting
$l_{virtual}$	=	light source direction for virtual lighting
ρ_{proto}	=	albedo for the prototype face
ρ_{nov}	=	albedo for the novel face

The prior knowledge of the lighting transformation can be represented by the ratio of the prototype image intensities under the two lighting directions

$$\frac{\rho_{proto}(\eta \cdot l_{virtual})}{\rho_{proto}(\eta \cdot l_{std})}.$$

Simply by multiplying by the image intensity of the novel person $\rho_{nov}(\eta \cdot l_{std})$ and cancelling terms, one can get

$$\rho_{nov}(\eta \cdot l_{virtual}),$$

which is the image intensity of the novel feature point under the virtual lighting. Overall, the novel face texture is modulated by the changes in the prototype lighting, an approach that has been explored by Brunelli [27].

Expression

In this case, the prototypes are fixed in pose and lighting but differ in expression, with the standard view being, say, a neutral expression and the virtual view being a smile, frown, etc. When generating virtual views, we need to capture both nonrigid shape deformations and the subtle texture changes such as the darkening effect of dimples or winkles. Thus, virtual views generation techniques for both shape and texture are required.

Predicting virtual expressions, however, seems more difficult than the rotation or lighting case. This is because the way a person smiles or frowns is probably decoupled from how to decompose his neutral face as a linear combination of the prototypes. To the extent that they are decoupled, the approaches we have suggested for generating virtual shapes and textures will be an approximation. Our problems show up mathematically in the nonrigidness of the transformation; the linear class idea for shape assumes a rigid 3D transform. The implication of these problems is that the expense of multiple prototypes is probably not justified; one is probably better off using just one or a few prototypes. In earlier work aimed primarily at computer graphics [19], we demonstrated parallel deformation for transformations from neutral to smiling expressions.

7.6.4 Future work

For future work on our approach to virtual views, we plan to use multiple prototypes for generating virtual shape. Vetter and Poggio [132] have already done some work in applying the linear class idea to both shape and texture. It would be interesting to test some of their virtual views in a view-based recognizer. In the longer term, one can test the virtual views technique for face recognition under different lighting conditions or expressions.

7.7 Summary

In this chapter we have addressed the problem of recognizing faces under different poses when only one example view of each person is available. Given one real view at a known pose, we use prior knowledge of faces to generate *virtual views*, views of the face as seen from different poses. Rather than using a more traditional 3D modeling approach, prior knowledge of faces is expressed in the form of 2D views of rotating prototype faces. Given the 2D prototype views and a single real view of a novel person, we demonstrated two techniques for effectively rotating the novel face in depth. First, in parallel deformation, a facial transformation observed on a prototype face in mapped onto a novel face and used to warp the novel view. Second, in linear classes, the single novel view is decomposed as a linear combination of prototype views at the same pose. Then these same linear coefficients are used to synthesize a virtual view of the novel person by taking a linear combination of the prototype views at virtual pose. We demonstrated this for the grey level, or textural, component of the face.

To evaluate virtual views, they were then used as example views in a view-based, pose-invariant face recognizer. On a database of 62 people with 10 test views per person, a recognition rate of 85% was achieved in experiments with parallel deformation, which is well above the base recognition rate of 70% when only one real view (plus its mirror reflection) is used. Also, our recognition rate is similar to other face recognition experiments where extrapolation from the pose-expression range of the example views is tested. Overall, for the problem of generating new views of an object from just one view, these results demonstrate that the 2D example-based technique, similarly to 3D object models, may be a viable method for representing knowledge of object classes.

Chapter 8

Discussion

8.1 Summary

This thesis has addressed the problem of automatic face recognition, the task of visually identifying a person in an input image. While this problem has been studied in the computer vision and pattern recognition communities for over two decades, most existing work in face recognition has limited the scope of the problem by dealing primarily with frontal views, neutral expressions, and fixed lighting conditions. To help generalize existing face recognition systems, this thesis has looked at the problem of recognizing faces under a range of viewpoints. The difficult part of this is to handle the two rotations out of the image plane, or rotations "in depth". In particular, we considered two cases of this problem:

1. many example views are available of each person, and

2. only one view is available per person.

In the latter case, perhaps the single available view per person is from a driver's license or passport photograph.

8.1.1 Multiple views per person

In the multiple views case, a simple view-based approach is taken to build a pose-invariant face recognition system. Each person in the database is represented using 15 views that sample a range of viewpoints on the viewing sphere. The 15 views include 5 rotations left/right (covering the range $[-30°, 30°]$) and 3 rotations up/down (covering the range $[-20°, 20°]$). Each view is represented using templates of the eyes, nose, and mouth, which is motivated by the prior success of template-based recognition systems for frontal views of faces [11][28][59].

Given a new input image to recognize, the view-based recognizer follows a basic strategy of first geometrically registering the input image with stored example views and then using normalized correlation to evaluate the match. To perform the geometrical registration, the first step is to automatically locate the two irises and a nose feature in the input. This feature detection module needs to be both person- and pose-invariant since it is the first thing that is run in the system. To satisfy these requirements, we use a simple template-based strategy for locating the eyes and nose, using hundreds of templates of the eyes-nose region from a variety of "exemplar" people and different poses. The feature detection problem becomes one of finding a good match between the input and one of the eyes-nose templates. To help keep the amount of computation under control, a hierarchical coarse-to-fine template-matching strategy is used.

Once the eyes and nose features are located, the input image is registered and matched against the example views in the face database. Based on a coarse pose estimate of the out-of-plane image rotation from the feature finder, only 9 views of the original 15 views are tested for each person. The matching procedure between the input image and a particular example view is as follows. First, the input image is registered with the example view in two steps

1. *Coarse registration using an affine transform.* The input image is resampled under an affine transform to align the eyes and nose features in the input image with the same features in the example view.

2. *Fine registration using optical flow.* Pixelwise correspondence is established between the affine-transformed input and the example view using optical flow. Then the affine-transformed input is warped using the correspondences to register the two images at the pixel-level.

The next step is to correlate the eyes, nose, and mouth templates from the example view against the registered input image. A normalized correlation metric is used to help provide some invariance to differences in lighting conditions between the two images. After performing this matching procedure for all 9 example views per person, the recognizer returns the best match.

When evaluated on a database of 62 people and 10 test views per person, our view-based recognizer attains a recognition rate of 98%. The test views cover the same range of viewpoints as the 15 example views, and the poses are randomly chosen by each person in the database. In addition, half of the test views include a rotation in the image plane, so all three rotational degrees of freedom were tested. The lighting conditions in the database are fixed, and people are asked to show a neutral expression.

8.1.2 Single view per person

In the second case of pose-invariant face recognition, we assume that only a single example view of each person is available in the database. In this case, is it still possible to recognize these people under a variety of poses, especially when new input views differ from the single available example by a rotation in depth? In this thesis, we reduced this case to the multiple views case by synthesizing *virtual views* [106][110]. Virtual views are new views of an object as seen from different poses, lighting conditions, or expressions. For our problem of pose-invariant face recognition, we are interested in virtual views under different rotations in depth.

How does one synthesize virtual views of an object? If the object belongs to a specific class of objects – such as the class of faces – then one may be able to take advantage of modeling assumptions on the class level to synthesize virtual views. That is, if one has prior knowledge about the object class, then one may be able to apply that knowledge to a single view of a "novel" object to synthesize virtual views of it. For instance, if one knows how faces change appearance under some transformation (e.g. rotation in depth, expression change), then that transformation can be applied to the single view available of a "novel" face. In this thesis, prior knowledge of face rotation in depth was encoded by using 2D views of prototype faces. Let the pose of the single novel object be defined as standard pose, and let the pose of the desired virtual view be virtual pose. Then the required views of the prototype faces are at both standard and virtual poses.

Two techniques were presented for synthesizing virtual views. First, the technique of *parallel deformation* (also see [108]) maps a transformation observed on a prototype object onto the novel object. The prior knowledge of the transformation is represented by a 2D deformation of feature locations on the prototype face. This 2D deformation records feature correspondence between the standard and virtual poses of the prototype, and we measure it using optical flow. Next, the prototype deformation is mapped onto the novel object, which requires feature correspondence between the standard poses of the prototype and novel objects. In this thesis, we explored both manual and automatic techniques for establishing these feature correspondences, the latter of which is based on our face "vectorizer". Finally, the mapped prototype deformation is used to 2D warp the novel object to synthesize the virtual view. Overall, the accuracy of the virtual view depends on the degree to which the 3D shapes of the prototype and novel objects match.

The second technique for synthesizing virtual views uses the concept of *linear classes* (see [106][110]). The main idea behind linear classes is that the novel object can be decomposed as a linear combination of a set of prototype objects. This decomposition is performed separately for object shape (locations of a set of feature points) and object texture (grey level values at a set of feature points). To synthesize a virtual view, one first computes the linear decomposition at the standard pose, which produces a set of linear coefficients. These coefficients are then transferred to the virtual pose, where virtual shape and texture of the novel object are computed by taking proper linear combinations of the shapes and textures of the prototypes. In this thesis, we explored this idea for synthesizing virtual textures.

The virtual views generated using parallel deformation and linear classes were evaluated by plugging them into our view-based recognizer for pose-invariant face recognition. Starting with a database with just one view per person, virtual views were used to augment the database to 15 views per person, the number of views used in the original view-based recognizer. The resulting face recognizer was tested on the same test set as with the real views case: 62 people, 10 views per person. Parallel deformation performed better than linear classes, achieving a recognition rate of 85%. To compare this recognition rate against a "base case", we ran another test of the view-based recognizer using only two views per person: the standard pose plus its mirror reflection. This yielded a recognition rate of only 70%. Thus, virtual views brought the recognition rate roughly halfway from the base case of 70% to the best case of 98% (using real views).

8.2 Contributions

8.2.1 Main contributions

The main contributions stem from the two subcases of pose-invariant face recognition: multiple views available and only one view available.

1. *View-based approach for pose-invariant face recognition.* When multiple example views from the viewing sphere are available of each person, we demonstrated that a simple view-based approach can be taken for the problem of pose-invariant face recognition. Our view-based face recognizer is the first face recognizer to handle a wide range of angles for all three rotational degrees of freedom. When comparing the input to stored example views, it follows a simple register-and-correlate strategy.

2. *Virtual views.* By showing that the addition of virtual views can boost the recognition rate of a view-based face recognizer, we demonstrated the usefulness of the concept of virtual views. In general, virtual views may be useful for using prior knowledge of object class to leverage a small example set. Besides object

recognition, this may be helpful for increasing the number of training examples in a learning-from-examples framework.

8.2.2 Secondary contributions

The secondary contributions stem from the feature finding requirements of the view-based recognizer and our techniques for synthesizing virtual views.

1. *Person- and pose-independent feature finder.* We developed a template-based system for automatically locating the two irises and a nose feature. The system works under a range of rotation angles both in and out of the image plane. While it was used in this thesis as a preliminary step in the face recognizer, it could also be used for applications like human-computer interaction or low-bandwidth videoconferencing, if only to initialize a tracking system.

2. *Face "vectorizer".* The face vectorizer is an automatic and person-independent technique for (i) locating a dense set of facial features, and (ii) modeling the grey levels of the face as a linear combination of prototype faces. The vectorizer works by exploiting a positive feedback loop between the feature correspondence process and the grey-level texture model which uses linear combinations. While the primary use of the vectorizer is to find feature correspondence between two arbitrary faces, the representation returned by the vectorizer is fairly general and could be used for tasks such as feature detection, expression analysis, and face recognition.

8.3 Future work

8.3.1 Analysis by synthesis

In dealing with varying pose, our face recognizer uses a simple view-based approach that stores many example views per person. Recognizing an input view boils down to retrieving an example view that is a close match. One problematic feature of this approach happens when one tries to recognize an input view whose pose falls midway between the poses of the closest example views. Our current system tries to compensate for this by resampling the input under an affine transform and warping the result using optical flow. The latter optical flow step, however, is ad hoc and could be improved.

A cleaner but slightly more complicated approach for the pose-invariant recognition problem is *analysis by synthesis.* The basic idea behind this approach is to try to resynthesize the input view using a face synthesis module. On an abstract level, one can think of the face synthesis module as a parameterized face model

$$M(\mathbf{p}) = \texttt{face-model (p)}$$

that can assume different facial appearances with different settings of the multidimensional vector $\mathbf{p}$. The elements of $\mathbf{p}$ may include parameters like rotation angles, expression parameters, or eigenface coefficients. The process of recognizing an input view I or estimating its pose can now be phrased as an optimization problem: try to find the best $\mathbf{p}$ that minimizes the difference $\|I - M(\mathbf{p})\|^2$.

A simple iteration between analysis and synthesis is one technique for solving for the vector $\mathbf{p}$. Pseudocode for a generalized version of the algorithm is as follows.

```
algorithm: analysis-by-synthesis
    input: image $I$
    output: parameter vector $\mathbf{p}$

    (1) $\mathbf{p}$ = initial estimate
    (2) loop until $\mathbf{p}$ stabilizes
    (3)       $M(\mathbf{p})$ = face-model (p)
    (4)       measure difference $D = I - M(\mathbf{p})$
    (5)       use $D$ to update $\mathbf{p}$
```

The algorithm maintains an estimate for $\mathbf{p}$, which at the beginning needs to be initialized with some good guess. In the analysis-synthesis loop, first the face model is used to synthesize $M(\mathbf{p})$, which hopefully is close to the input I being analyzed. Next, the difference between I and $M(\mathbf{p})$ is measured and used to update the parameter vector $\mathbf{p}$ so that $M(\mathbf{p})$ is brought closer to I in the next iteration. When this iterative approach is tested in the future, there is the issue of convergence to be addressed. The convergence properties of the algorithm are probably dependent on factors such as the initial conditions for $\mathbf{p}$ and the convexity of the functional $\|I - M(\mathbf{p})\|^2$.

Since we are interested in the problem of recognition under varying pose, the parameter vector $\mathbf{p}$ should include the rotation angles out of the image plane (r_1, r_2). Given our current database, the synthesis module would interpolate between the 15 example views of each person to synthesize intermediate poses. As shown in Fig. 8-1, the space of out-of-plane rotations could be divided into 8 cells, with each cell containing four views. Within a particular cell, the face model would interpolate the four views as a function of (r_1, r_2), the amount of rotation up/down and left/right. Thus, instead of looping over all the views as in the current view-based approach, the new analysis-synthesis system tries to find an optimal (r_1, r_2) by interpolating the examples.

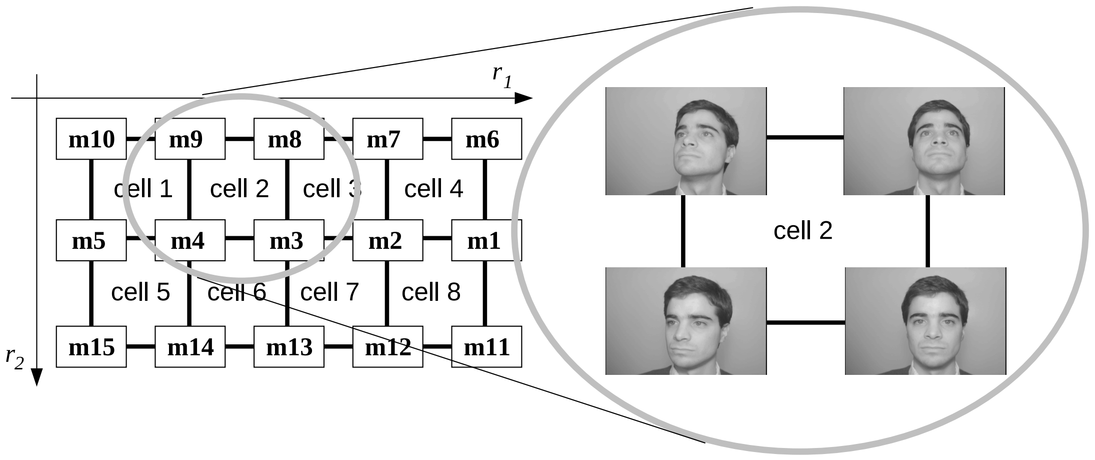


Figure 8-1: In our proposed analysis-synthesis system for varying pose, we interpolate between the 15 example views of each person. The 15 views are divided into 8 cells of 4 views each, and $\mathbf{p} = (r_1, r_2)$.

This analysis by synthesis approach can be applied in three different areas of this thesis.

1. *Recognition using multiple views.* This is the scenario where all 15 views per person are available in the database. The current view-based system compares an input against a particular person by iteratively matching the input against 9 of the 15 views. The new analysis-synthesis system would first interpolate the views to try to reconstruct the input. Then this reconstruction would be correlated against the input.

2. *Multi-view vectorizer.* The current vectorizer is tuned to the particular out-of-plane rotation of the prototype "training" views. It should be possible to link a set of vectorizers as shown in Fig. 8-1 by defining correspondence between the standard shapes of vectorizers in the same cell. In this new multi-view vectorizer, the parameter vector $\mathbf{p}$ would not only include (r_1, r_2), but also the linear texture coefficients from the vectorizer β_i. The similarity transform P and correspondences $\mathbf{y}_{a-std}^{std}$ would be auxiliary variables. For correspondence, there would be a set of standard shapes, one standard shape per cell.

3. *Recognition using one view.* This is the scenario addressed by the second part of the thesis, where only one view of each person is available in the database. Instead of synthesizing a set of virtual views off-line before recognition, a single

virtual view is synthesized at run-time for each person. The face model in this case synthesizes views using (a) the single view of each database person and (b) "prior knowledge" of face rotation from the prototype views. Hence, the techniques we discussed for synthesizing virtual views are bundled with the face synthesis module. A similar approach called the *visualization* route to recognition was discussed in Vetter, Hurlbert, and Poggio [131]. First, the pose of the face is estimated, and then prototype knowledge is used to normalize the input for out-of-plane rotations. The normalized input is then compared against the single view of each person in the database.

8.3.2 Linear classes without virtual views

For the problem of recognizing faces from just one example view, it should be possible to use the idea of linear classes without actually synthesizing virtual views. Consider the following proposal. Assume that we had a face vectorizer that computed a shape and a textural decomposition of an input image img_a in terms of linear combinations of prototype faces $\{img_{p_j}\}_{j=1}^N$

$$\mathbf{y}_a = \sum_{j=1}^N \alpha_j \mathbf{y}_{p_j} \qquad \mathbf{t}_a = \sum_{j=1}^N \beta_j \mathbf{t}_{p_j}.$$

This was discussed in section 6.6.2 as a possible future direction for the existing face vectorizer. Since the vectorizer is tuned for a specific view, let us assume that we have a collection of view-tuned vectorizers *each built using the same set of prototype people for each view.* The latter constraint will allow us to use ideas from linear classes to relate linear coefficients across different views.

The basic idea behind linear classes without virtual views is to compare faces based on sets of (α_j, β_j) coefficients rather than using correlation in an image space. Given the single real view per person at standard pose, we vectorize each view and store the set of coefficients $(\alpha_j, \beta_j)^{std}$ for each person in the database. When recognizing a new view at run-time, we vectorize the input by either

1. estimating the pose and choosing the correct view-tuned vectorizer, or

2. building a multi-pose vectorizer (see above) that links a group of vectorizers. The vectorization parameters and out-of-plane rotation are simultaneously computed.

The result is a set of linear coefficients $(\alpha_j, \beta_j)^{input}$ which have been computed using a set of prototype views that match the pose of the input.

According to linear classes, the (α_j, β_j) decomposition for a specific individual should be invariant to pose. As explained in Chapter 7, linear classes is based on the assumption that the 3D shape vector of the input **Y** and the 3D texture vector **T** are linear combinations of the shapes and textures of prototype faces. Under certain conditions, the linear coefficients (α_j, β_j) of the 3D decomposition are computable from an arbitrary 2D view. Thus, the coefficients should be invariant to pose since they are derived from a 3D representation. It follows that the (α_j, β_j) coefficients should themselves be an effective representation for faces. The coefficients of the unidentified input view $(\alpha_j, \beta_j)^{input}$ can be directly matched against the database coefficients of each person at standard pose $(\alpha_j, \beta_j)^{std}$. Note that the linear coefficients are not a true invariant because the recognizer at run-time needs to have an estimate of the out-of-plane image rotation of the input.

8.4 Closing remarks

This thesis has studied the problem of recognizing faces under varying pose. By addressing the issue of pose, our immediate goal was to help bring face recognition one step closer to the ultimate goal of recognition under general imaging conditions. Beyond pose, the other major sources of variation that need to be addressed are lighting conditions and expressions. But even if building a system that handles all these sources of variation proves elusive, there are still some applications where it is safe to assume restricted imaging conditions (e.g. verification for building access).

Beyond recognizing faces, there are many interesting tasks that involve processing images of faces. Take, for example, the problem of detecting faces in cluttered scenes or the problem of estimating facial parameters such as pose, expression, mouth articulation, and lighting conditions. Solutions to these problems will be be useful in applications like model-based coding for low-bandwidth videoconferencing, human-computer interaction, and performance-driven animation systems.

The success of our view-based face recognizer has an impact not only in the study of faces, but also lends some computational support to the use of the view-based approach in object recognition. Our experimental results supplements recent support for the view-based approach from psychophysics (Bülthoff, Edelman, and Tarr [29]), neurophysiology (Logothetis and Pauls [88]), and object recognition experiments (Poggio and Edelman [107]).

Within the larger context of object recognition, this thesis has addressed a discrimination task using a view-based approach. The recognition strategy developed in this thesis may be useful in other subordinate-level recognition tasks, such as the identification of animals like dogs and cats, or the identification of cars. One area that has not been addressed is the more general problem of basic-level recognition or categorization. The view-based approach may be useful here as well, but so may other approaches such as parts-based representations or 3D models. More study certainly needs to be done in this area, and there is probably not just one answer. In tackling the recognition problem, the human brain could use a variety of representations and approaches; we would be surprised if the view-based approach were not one of them.

Appendix A

Face Database

The face database contains 62 people, 25 views per person. So that we can explore the issue of pose, the different views of each person cover a variety of poses, including rotation angles both in and out of the image plane.

The database is divided into two parts, a set of example views and a set of testing views.

1. *Example views.* In our view-based approach for face recognition under varying pose, faces are represented using 15 example images that cover the viewing sphere. Shown in Fig. A-1, these views sample 5 left/right rotations and 3 up/down rotations. When a subject is added to the database of faces, example and test image data is taken with a camera perched on top of a workstation monitor. To help collect the example views, we fit a large piece of foam core around the monitor with dots indicating the viewing sphere locations being sampled. When taking the example views, the subject is asked to rotate his head to point his nose at each of the 15 dots. No mechanisms are used to make the subjects poses accurate relative to the ideal "dot" poses other than our oral instructions fine tuning the subject's pose. This field of dots sample the 5 left/right rotations at approximately -30, -15, 0, 15, and 30 degrees and the 3 up/down rotations at approximately -20, 0, and 20 degrees. The two rotation parameters are restricted so that the two eyes are always visible; this is why the left/right rotation parameter is not sampled beyond 30 degrees.

2. *Test views.* In addition to the 15 example views, 10 test views are taken per person. For these test views, the subject is instructed to choose 10 points at random within the rectangle defined by the outer border of dots. The test poses can fall close to example poses or in between them. The 10 views are divided into two groups of 5. The first group is similar to the example views in that only the left/right and up/down rotational parameters are allowed to vary. For the second group of 5, the subject is allowed to introduce image-plane rotation. See Fig. A-2 for example test views.

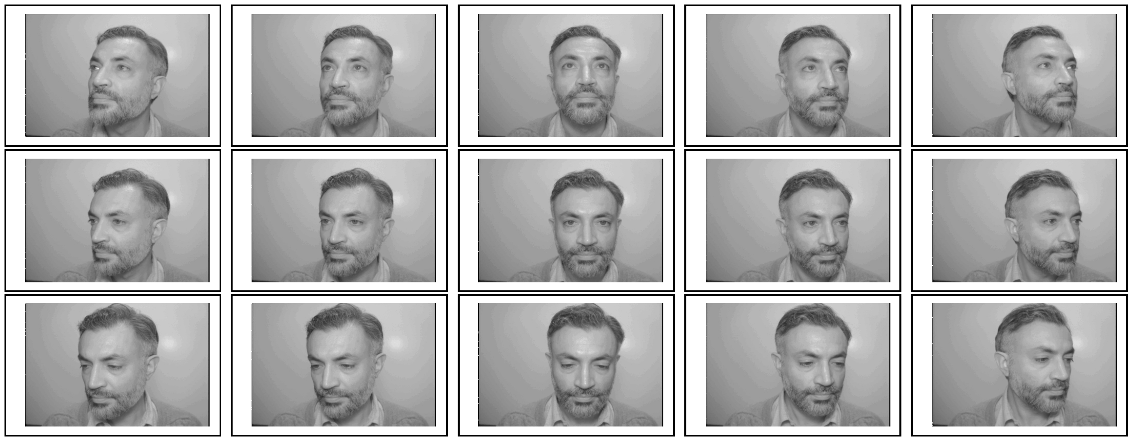


Figure A-1: The view-based face recognizer uses 15 example views per person.

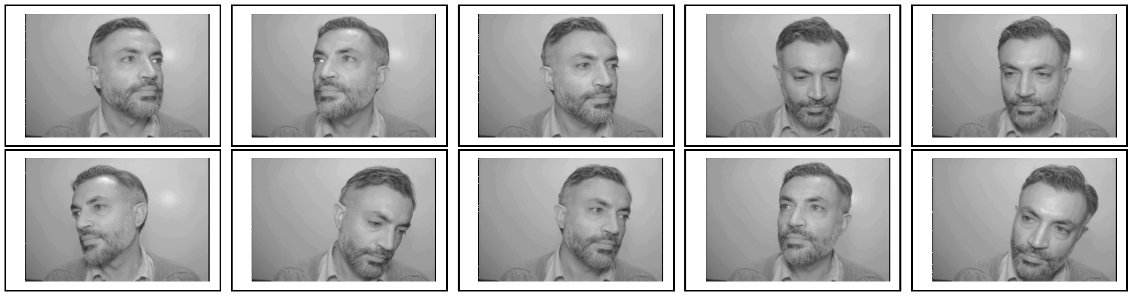


Figure A-2: For each person, 10 test images are taken that sample random poses from the viewing sphere.

We currently have 62 people in the database for a total of 930 example and 620 testing views. The collection of people is fairly varied, including 44 males and 18 females, people from different races, and an age range from the 20s to the 40s. The frontal views of everyone in the database are shown in Figs. A-3 and A-4.

For both the example and test views, the lighting conditions are fixed and consist of a 60 watt lamp near the camera supplemented by background lighting from windows and overhead lights. Facial expression is also fixed at a neutral expression.

After taking the example and test images, we manually specify the locations of the two irises, nose lobes, and corners of the mouth (see Fig. A-5). These manual feature locations are used for four purposes:

1. During batch evaluations of the feature finder, they serve as ground truth data for validating the locations returned by the feature finder.

Figure A-3: An exhaustive listing of people in the database, part 1 of 2.

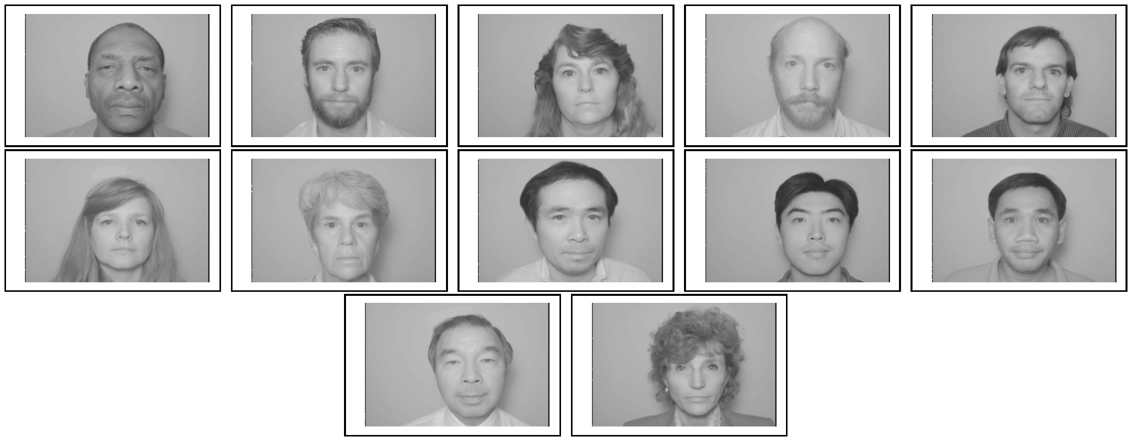


Figure A-4: An exhaustive listing of people in the database, part 2 of 2.

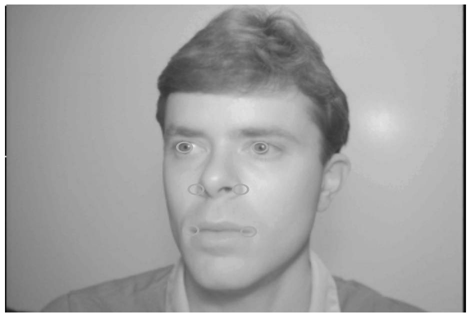


Figure A-5: The irises, nose lobes, and corners of the mouth are manually labeled for each image in the database.

2. Also in the feature finder, the manual locations define the exact (x, y) locations of the irises and nose lobes features within the eyes-nose templates. As explained in Chapter 3, these (x, y) locations are mapped to the input image using correspondences from optical flow in order to locate the irises and nose lobes in the input image.

3. For the recognizer itself, the feature locations are used to automatically define the bounding boxes of facial feature templates in the example images, as is discussed in Chapter 4.

4. Lastly, during the geometrical alignment step between input and example images, the recognizer registers the automatically located input features to the

manually located example image features.

Appendix B

Linear Classes: Shape and Texture

As explained in section 7.3.1, linear classes is a technique for synthesizing new views of an object using views of prototypical objects belonging to the same object class. The basic idea is to decompose the novel object as a linear combination of the prototype objects. This decomposition is performed separately for the shape and texture of the novel object. In this appendix, we explain the mathematical detail behind the linear class approach for shape and texture. Please refer to sections 7.2 and 7.3.1 for definitions of the example prototype images, mathematical operators, etc.

B.1 Shape

In this section, we reformulate the description of linear classes for shape that originally appeared in Poggio and Vetter [110]. The development here makes explicit the fact that the vectorized $\mathbf{y}$ vectors need not be in correspondence between the standard and virtual poses.

Linear classes begins with the assumption that a novel object is a linear combination of a set of prototype objects in 3D

$$\mathbf{Y}_n = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{Y}_{p_j}. \tag{B.1}$$

From this assumption, it is easy to see that any 2D view of the novel object will be the same linear combination of the corresponding 2D views of the prototypes. That is, the 3D linear decomposition is the same as the 2D linear decomposition. Using equation (7.2) which relates 3D and 2D shape vectors, let $\mathbf{y}_{n,r}$ be a 2D view of a novel object

$$\mathbf{y}_{n,r} = L\mathbf{Y}_n \tag{B.2}$$

and let $\mathbf{y}_{p_j,r}$ be 2D views of the prototypes

$$\mathbf{y}_{p_j,r} = L\mathbf{Y}_{p_j} \qquad 1 \le j \le N. \tag{B.3}$$

Apply the operator L to both sides of equation (B.1)

$$L\mathbf{Y}_n = L(\textstyle\sum_{j=1}^{N} \alpha_j \mathbf{Y}_{p_j}). \tag{B.4}$$

We can bring L inside the sum since L is linear

$$L\mathbf{Y}_n = \textstyle\sum_{j=1}^{N} \alpha_j L\mathbf{Y}_{p_j}. \tag{B.5}$$

Substituting equations (B.2) and (B.3) yields

$$\mathbf{y}_{n,r} = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{y}_{p_j,r}.$$

Thus, the 2D linear decomposition uses the same set of linear coefficients as with the 3D vectorization.

Next, we show that under certain assumptions, the novel object can be analyzed at standard pose and the virtual view synthesized at virtual pose using a single set of linear coefficients. Again, assume that a novel object is a linear combination of a set of prototype objects in 3D

$$\mathbf{Y}_n = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{Y}_{p_j}. \tag{B.6}$$

Say that we have 2D views of the prototypes at standard pose $\mathbf{y}_{p_j}$, 2D views of the prototypes at virtual pose $\mathbf{y}_{p_j,r}$, and a 2D view of the novel object $\mathbf{y}_n$ at standard pose. Additionally, assume that the 2D views $\mathbf{y}_{p_j}$ are linearly independent. Project both sides of equation (B.6) using the rotation for standard pose, yielding

$$\mathbf{y}_n = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{y}_{p_j}.$$

A unique solution for the α_j exist since the $\mathbf{y}_{p_j}$ are linearly independent. Now, since we have solved for the same set of coefficients in the 3D linear class assumption, the decomposition at virtual pose must use the same coefficients

$$\mathbf{y}_{n,r} = \textstyle\sum_{j=1}^{N} \alpha_j \mathbf{y}_{p_j,r}.$$

That is, we can recover the α_j's from the view at standard pose and use the α_j's to generate the virtual view of the novel object.

B.2 Texture

Virtually the same argument can be applied to the geometrically normalized texture vectors $\mathbf{t}$. The idea of applying linear classes to texture was thought of by the author and independently by Vetter and Poggio [132].

With the texture case, assume that a novel object texture $\mathbf{T}_n$ is a linear combination of a set of prototype textures

$$\mathbf{T}_n = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{T}_{p_j}. \tag{B.7}$$

As with shape, we show that the 3D linear decomposition is the same as the 2D linear decomposition. Using equation (7.3) which relates 3D and 2D texture vectors, let $\mathbf{t}_{n,r}$ be a 2D texture of a novel object

$$\mathbf{t}_{n,r} = D\mathbf{T}_n \tag{B.8}$$

and let $\mathbf{t}_{p_j,r}$ be 2D textures of the prototypes

$$\mathbf{t}_{p_j,r} = D\mathbf{T}_{p_j} \qquad 1 \leq j \leq N. \tag{B.9}$$

Apply the operator D to both sides of equation (B.7)

$$D\mathbf{T}_n = D(\textstyle\sum_{j=1}^{N} \beta_j \mathbf{T}_{p_j}). \tag{B.10}$$

We can bring D inside the sum since D is linear

$$D\mathbf{T}_n = \textstyle\sum_{j=1}^{N} \beta_j D\mathbf{T}_{p_j}. \tag{B.11}$$

Substituting equations (B.8) and (B.9) yields

$$\mathbf{t}_{n,r} = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j,r}.$$

Thus, as with shape, the 2D linear decomposition for texture uses the same set of linear coefficients as with the 3D vectorization.

Next, we show that under certain linear independence assumptions, the novel object texture can be analyzed at standard pose and the virtual view synthesized at virtual pose using a single set of linear coefficients. Again, assume that a novel object texture $\mathbf{T}$ is a linear combination of a set of prototype objects

$$\mathbf{T}_n = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{T}_{p_j}. \tag{B.12}$$

Say that we have 2D textures of the prototypes at standard pose $\mathbf{t}_{p_j}$, the 2D prototype textures at virtual pose $\mathbf{t}_{p_j,r}$, and a 2D texture of the novel object at standard pose $\mathbf{t}_n$. Additionally, assume that the 2D textures $\mathbf{t}_{p_j}$ are linearly independent. Project both sides of equation (B.12) using the rotation for standard pose, yielding

$$\mathbf{t}_n = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j}.$$

A unique solution for the β_j exist since the $\mathbf{t}_{p_j}$ are linearly independent. Now, since we have solved for the same set of coefficients in the 3D linear class assumption, the decomposition at virtual pose must use the same coefficients

$$\mathbf{t}_{n,r} = \textstyle\sum_{j=1}^{N} \beta_j \mathbf{t}_{p_j,r}.$$

That is, we can recover the β_j's from the view at standard pose and use the β_j's to generate the virtual view of the novel object.

Bibliography

[1] Y. Abu-Mostafa. A method for learning from hints. In S. J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural information processings systems 5*, pages 73–80, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

[2] Y.S. Abu-Mostafa. Hints and the VC-dimension. *Neural Computation*, 5:278–288, 1993.

[3] Andrew C. Aitchison and Ian Craw. Synthetic images of faces – an approach to model-based face recognition. In *Proc. British Machine Vision Conference*, pages 226–232, 1991.

[4] K. Aizawa, H. Harashima, and T. Saito. Model-based analysis synthesis image coding (MBASIC) system for a person's face. *Signal Processing: Image Communication*, 1:139–152, 1989.

[5] Shigeru Akamatsu, Tsutomu Sasaki, Hideo Fukamachi, Nobuhiko Masui, and Yasuhito Suenaga. An accurate and robust face identification scheme. In *Proceedings Int. Conf. on Pattern Recognition*, volume 2, pages 217–220, The Hague, The Netherlands, 1992.

[6] Takaaki Akimoto, Yasuhito Suenaga, and Richard S. Wallace. Automatic creation of 3D facial models. *IEEE Computer Graphics and Applications*, 13(5):16–22, 1993.

[7] Scott E. Anderson and Mark A.Z. Dippe. A hybrid approach to facial animation. Technical Report 1026, Industrial Light and Magic, San Rafael, California, January 1990.

[8] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. Technical Report No. 180, MIT Media Lab, Vision and Modeling Group, 1992.

[9] Henry S. Baird. *Model-Based Image Matching Using Location.* The MIT Press, Cambridge, MA, 1985.

[10] Dana H. Ballard and Chrisopher M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[11] Robert J. Baron. Mechanisms of human facial recognition. *International Journal of Man Machine Studies*, 15:137–178, 1981.

[12] Adam Baumberg and David Hogg. Learning flexible models from image sequences. In *Proceedings of the European Conference on Computer Vision*, pages 299–308, Stockholm, Sweden, 1994.

[13] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92 Proceedings*, pages 35–42, Chicago, IL, 1992.

[14] Jezekiel Ben-Aire and K. Raghunath Rao. On the recognition of occluded shapes and generic faces using multiple-template expansion matching. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 214–219, New York, NY, 1993.

[15] Alan Bennett and Ian Craw. Finding image features using deformable templates and detailed prior statistical knowledge. In *Proc. British Machine Vision Conference*, pages 233–239, 1991.

[16] James R. Bergen and Edward H. Adelson. Hierarchical, computationally efficient motion estimation algorithm. *J. Opt. Soc. Am. A*, 4(13):P35, 1987.

[17] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pages 237–252, Santa Margherita Ligure, Italy, June 1992.

[18] J.R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, Princeton, New Jersey, April 1990.

[19] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. A.I. Memo No. 1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[20] Martin Bichsel. *Strategies of Robust Object Recognition for the Automatic Identification of Human Faces*. PhD thesis, ETH, Zurich, 1991.

[21] Andrew Blake and Michael Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *SIGGRAPH '94 Proceedings*, pages 185–192, Orlando, FL, 1994.

[22] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: The local–feature–focus method. *International Journal of Robotics Research*, 1(3):57–82, 1982.

[23] Thomas M. Breuel. Geometric aspects of visual object recognition. Technical Report AI–TR 1374, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[24] Thomas M. Breuel. An efficient correspondence based algorithm for 2D and 3D model based recognition. A.I. Memo No. 1259, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[25] Vicki Bruce. Changing faces: Visual and non-visual coding processes in face recognition. *British Journal of Psychology*, 73:105–116, 1982.

[26] R. Brunelli and T. Poggio. Hyberbf networks for real object recognition. In *Proceedings IJCAI*, Sydney, Australia, 1991.

[27] Roberto Brunelli. Estimation of pose and illuminant direction for face processing. A.I. Memo No. 1499, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1994.

[28] Roberto Brunelli and Tomaso Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[29] Heinrich H. Bülthoff, Shimon Y. Edelman, and Michael J. Tarr. How are three-dimensional objects represented in the brain? *Cerebral Cortex*, 3:247–260, May/June 1995.

[30] J. Brian Burns, Richard Weiss, and Edward M. Riseman. View variation of point set and line segment features. In *Proceedings Image Understanding Workshop*, pages 650–659, Pittsburgh, PA, September 1990.

[31] Peter J. Burt. Smart sensing within a pyramid vision machine. *Proceedings of the IEEE*, 76(8):1006–1015, August 1988.

[32] Peter J. Burt. Multiresolution techniques for image representation, analysis, and 'smart' transmission. In *SPIE Vol. 1199, Visual Communications and Image Processing IV*, pages 2–15, 1989.

[33] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, COM-31(4):532–540, April 1983.

[34] J.C. Campos, A.D. Linney, and J.P. Moss. The analysis of facial profiles using scale space techniques. *Pattern Recognition*, 26(6):819–824, 1993.

[35] Scott R. Cannon, Gregory W. Jones, Robert Campbell, and Neil W. Morgan. A computer vision system for identification of individuals. In *Proc. IECON*, pages 347–351, Milwaukee, WI, 1986.

[36] Todd A. Cass. Polynomial-time object recognition in the presence of clutter, occlusion, and uncertainty. In *Proceedings of the European Conference on Computer Vision*, pages 834–842, 1992.

[37] Indranil Chakravarty and Herbert Freeman. Characteristic views as a basis for three-dimensional object recognition. In *SPIE Vol. 336, Robot Vision*, pages 37–45, 1982.

[38] Chin-Wen Chen and Chung-Lin Huang. Human face recognition from a single front view. *International Journal of Pattern Recognition and Artificial Intelligence*, 6(4):571–593, 1992.

[39] Yong-Qing Cheng, Ke Liu, Jing-Yu Yang, and Hua-Feng Wang. A robust algebraic method for human face recognition. In *Proceedings Int. Conf. on Pattern Recognition*, volume 2, pages 221–224, The Hague, The Netherlands, 1992.

[40] Chang Seok Choi, Hiroshi Harashima, and Tsuyosi Takebe. Analysis and synthesis of facial expressions in knowledge-based coding of facial image sequences. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2737–2740, 1991.

[41] David Clemens and David Jacobs. Model-group indexing for recognition. In *Proceedings Image Understanding Workshop*, pages 604–613, Pittsburgh, PA, September 1990.

[42] T.F. Cootes and C.J. Taylor. Active shape models - 'Smart snakes'. In David Hogg and Roger Boyle, editors, *Proc. British Machine Vision Conference*, pages 266–275. Springer Verlag, 1992.

[43] T.F. Cootes and C.J. Taylor. Using grey-level models to improve active shape model search. In *Proceedings Int. Conf. on Pattern Recognition*, volume 1, pages 63–67, Jerusalem, Israel, 1994.

[44] T.F. Cootes, C.J. Taylor, A. Lanitis, D.H. Cooper, and J. Graham. Building and using flexible models incorporating grey-level information. In *Proceedings of the International Conference on Computer Vision*, pages 242–246, Berlin, May 1993.

[45] Ian Craw and Peter Cameron. Parameterizing images for recognition and reconstruction. In *Proc. British Machine Vision Conference*, pages 367–370, 1991.

[46] Ian Craw and Peter Cameron. Face recognition by computer. In David Hogg and Roger Boyle, editors, *Proc. British Machine Vision Conference*, pages 498–507. Springer Verlag, 1992.

[47] Ian Craw, David Tock, and Alan Bennett. Finding face features. In *Proceedings of the European Conference on Computer Vision*, pages 92–96, 1992.

[48] Robert Desimone. Face-selective cells in the temporal cortex of monkeys. *Journal of Cognitive Neuroscience*, 3(1):1–8, 1991.

[49] Shimon Edelman, Daniel Reisfeld, and Yechezkel Yeshurun. Learning to recognize faces from examples. In *Proceedings of the European Conference on Computer Vision*, pages 787–791, 1992.

[50] P. Ekman and W.V. Friesen. *Facial Action Coding System.* Consulting Psychologists Press, Inc., Palo Alto, CA, 1977.

[51] Irfan A. Essa and Alex Pentland. A vision system for observing and extracting facial action parameters. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 76–83, Seattle, WA, 1994.

[52] Nancy L. Etcoff, Roy Freeman, and Kyle R. Cave. Can we lose memories of faces? content specificity and awareness in a prosopagnosic. *Journal of Cognitive Neuroscience*, 3(1):25–41, 1991.

[53] Michael K. Fleming and Garrison W. Cottrell. Categorization of faces using unsupervised feature extraction. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 65–70, 1990.

[54] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant descriptors for 3-D object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, 1991.

[55] A. Fuchs and H. Haken. Pattern recognition and associative memory as dynamical processes in a synergetic system; I. translational invariance, selective attention, and decomposition of scenes. *Biological Cybernetics*, 60:17–22, 1988.

[56] A. Fuchs and H. Haken. Pattern recognition and associative memory as dynamical processes in a synergetic system; II. decomposition of complex scenes, simultaneous invariance with respect to translation, rotation, scaling. *Biological Cybernetics*, 60:107–109, 1988.

[57] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[58] Ziv Gigus and Jitendra Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):113–122, February 1990.

[59] Jeffrey M. Gilbert and Woody Yang. A real-time face recognition system using custom VLSI hardware. In *IEEE Workshop on Computer Architectures for Machine Perception*, pages 58–66, December 1993.

[60] Chris Goad. Special purpose automatic programming for 3D model-based vision. In *Proceedings Image Understanding Workshop*, pages 94–104, Arlington, VA, June 1983.

[61] Gaile G. Gordon. Face recognition based on depth and curvature features. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 808–810, 1992.

[62] W. Eric L. Grimson and Tomas Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):467–482, 1987.

[63] Peter W. Hallinan. Recognizing human eyes. In *SPIE Vol. 1570, Geometric Methods in Computer Vision*, pages 214–226, 1991.

[64] Peter W. Hallinan. A low-dimensional representation of human faces for arbitrary lighting conditions. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 995–999, Seattle, WA, 1994.

[65] J. Richard Hanley, Andrew W. Young, and Norma A. Pearson. Defective recognition of familiar people. *Cognitive Neuropsychology*, 6(2):179–210, 1989.

[66] L.D. Harmon, M.K. Khan, Richard Lasch, and P.F. Ramig. Machine identification of human faces. *Pattern Recognition*, 13(2):97–110, 1981.

[67] Zi-Quan Hong. Algebraic feature extraction of image for recognition. *Pattern Recognition*, 24(3):211–219, 1991.

[68] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[69] Chung-Lin Huang and Chin-Wen Chen. Human facial feature extraction for face interpretation and recognition. *Pattern Recognition*, 25(12):1435–1444, 1992.

[70] Daniel P. Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[71] Katsushi Ikeuchi and Takeo Kanade. Applying sensor models to automatic generation of object recognition programs. In *Proceedings of the International Conference on Computer Vision*, pages 228–237, Tampa, FL, December 1988.

[72] Michael J. Jones and Tomaso Poggio. Model-based matching of line drawings by linear combinations of prototypes. In *Proceedings of the International Conference on Computer Vision*, pages 531–536, Boston, Massachusetts, June 1995.

[73] Takeo Kanade. Picture processing by computer complex and recognition of human faces. Technical report, Kyoto University, Dept. of Information Science, 1973.

[74] Chii-Yuan Kang, Yung-Sheng Chen, and Wen-Hsing Hsu. Mapping a lifelike 2.5D human face via an automatic approach. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 611–612, New York, NY, June 1993.

[75] Gerald J. Kaufman and Kenneth J. Breeding. The automatic recognition of human faces from profile silhouettes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(2):113–121, February 1976.

[76] Y. Kaya and K. Kobayashi. A basic study on human face recognition. In Satosi Watanabe, editor, *Frontiers of Pattern Recognition*, pages 265–289. Academic Press, New York, NY, 1972.

[77] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[78] J. J. Koenderink and A. J. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[79] Jan J. Koenderink and Andrea J. van Doorn. Affine structure from motion. *J. Opt. Soc. Am. A*, 8(2):377–385, 1991.

[80] T. Kohonen. *Self-organization and Associative Memory.* Springer-Verlag, Berlin, 1989.

[81] Matthew R. Korn and Charles R. Dyer. 3-D multiview object representations for model-based object recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[82] T. Kurita, N. Otsu, and T. Sato. A face recognition method using higher order local autocorrelation and multivariate analysis. In *Proceedings Int. Conf. on Pattern Recognition*, volume 2, pages 213–216, The Hague, The Netherlands, 1992.

[83] Martin Lades, Jan C. Vorbruggen, Joachim Buhmann, Jorg Lange, Christoph v.d. Malsburg, Rolf P. Wurtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3), March 1993.

[84] Maria Lando and Shimon Edelman. Generalization from a single view in face recognition. In *Proceedings, International Workshop on Automatic Face- and Gesture-Recognition*, pages 80–85, Zurich, 1995.

[85] J.T. Lapreste, J.Y Cartoux, and M. Richetin. Face recongition from range data by structural analysis. In G. Ferrate and *et al.*, editors, *Syntactic and Structural Pattern Recognition*, pages 303–313. Springer-Verlag, Berlin, 1988.

[86] John C. Lee and Evangelos Milios. Matching range images of human faces. In *Proceedings of the International Conference on Computer Vision*, pages 722–726, Dec 1990.

[87] Peter Litwinowicz and Lance Williams. Animating images with drawings. In *SIGGRAPH '94 Proceedings*, pages 409–412, Orlando, FL, 1994.

[88] N.K. Logothetis and J. Pauls. Psychophysical and physiological evidence for viewer-centered object representations in the primate. *Cerebral Cortex*, 3:270–288, May/June 1995.

[89] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[90] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings IJCAI*, pages 674–679, Vancouver, 1981.

[91] B.S. Manjunath, R. Chellappa, and C. von der Malsburg. A feature based approach to face recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 373–378, 1992.

[92] B.S. Manjunath, Chandra Shekhar, R. Chellappa, and C. von der Malsburg. A robust method for detecting image features with application to face recognition and motion correspondence. In *Proceedings Int. Conf. on Pattern Recognition*, volume 2, pages 208–212, The Hague, The Netherlands, 1992.

[93] Thomas Maurer and Christoph von der Malsburg. Single-view based recognition of faces rotated in depth. In *Proceedings, International Workshop on Automatic Face- and Gesture-Recognition*, pages 248–253, Zurich, 1995.

[94] H. Midorikawa. The face pattern identification by back-propagation learning procedure. In *Abstracts of the First Annual INNS Meeting*, page 515, Boston, 1988.

[95] Tom M. Mitchell and Sebastian B. Thrun. Explanation-based neural network learning for robot control. In S. J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural information processings systems 5*, pages 287–294, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

[96] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Proceedings of the International Conference on Computer Vision*, pages 786–793, Cambridge, MA, June 1995.

[97] Yael Moses, Shimon Ullman, and Shimon Edelman. Generalization to novel images in upright and inverted faces. Technical Report CC93-14, The Weizmann Institute of Science, 1993.

[98] Hiroshi Murase and Shree K. Nayar. Learning object models from appearance. In *Proceedings AAAI*, pages 836–843, Washington, DC, 1993.

[99] Takashi Nagamine, Tetsuya Uemura, and Isao Masuda. 3D facial analysis for human identification. In *Proceedings Int. Conf. on Pattern Recognition*, volume 1, pages 324–327, The Hague, The Netherlands, 1992.

[100] Osamu Nakamura, Shailendra Mathur, and Toshi Minami. Identification of human faces based on isodensity maps. *Pattern Recognition*, 24(3):263–272, 1991.

[101] Masaaki Oka, Kyoya Tsutsui, Akio Ohba, Yoshitaka Kurauchi, and Takashi Tago. Real-time manipulation of texture-mapped surfaces. In *SIGGRAPH '87 Proceedings*, pages 181–188, Anaheim, CA, July 1987.

[102] Elizabeth C. Patterson, Peter C. Litwinowicz, and Ned Greene. Facial animation by spatial mapping. In *Computer Animation '91*, pages 31–44. Springer-Verlag, Tokyo, 1991.

[103] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 84–91, Seattle, WA, 1994.

[104] D.J. Perrett, P.A.J. Smith, D.D. Potter, A.J. Mistlin, A.S. Head, A.D. Milner, and M.A. Jeeves. Neurones responsive to faces in the temporal cortex: studies of functional organization, sensitivity to identity and relation to perception. *Human Neurobiology*, 3:197–208, 1984.

[105] T. Poggio. 3D object recognition: on a result by Basri and Ullman. Technical Report # 9005–03, IRST, Povo, Italy, 1990.

[106] T. Poggio. 3D object recognition and prototypes: one 2D view may be sufficient. Technical Report 9107–02, I.R.S.T., Povo, Italy, July 1991.

[107] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343(6255):263–266, January 1990.

[108] Tomaso Poggio and Roberto Brunelli. A novel approach to graphics. A.I. Memo No. 1354, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[109] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, Sept 1990.

[110] Tomaso Poggio and Thomas Vetter. Recognition and structure from one 2D model view: Observations on prototypes, object classes, and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[111] Jean Ponce and David J. Kriegman. Computing exact aspect graphs of curved objects: Parametric surfaces. In *Proceedings AAAI*, pages 1074–1079, 1990.

[112] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, 1988.

[113] C.S. Ramsay, K. Sutherland, D. Renshaw, and P.B. Denyer. A comparison of vector quantization codebook generation algorithms applied to automatic face recognition. In David Hogg and Roger Boyle, editors, *Proc. British Machine Vision Conference*, pages 508–517. Springer Verlag, 1992.

[114] Daniel Reisfeld, Nur Arad, and Yehezkel Yeshurun. Normalization of face images using few anchors. In *Proceedings Int. Conf. on Pattern Recognition*, volume 1, pages 761–763, Jerusalem, Israel, 1994.

[115] Daniel Reisfeld and Yehezkel Yeshurun. Robust detection of facial features by generalized symmetry. In *Proceedings Int. Conf. on Pattern Recognition*, volume 1, pages 117–120, The Hague, The Netherlands, 1992.

[116] Phillipe G. Schyns and Heinrich H. Bülthoff. Conditions for viewpoint invariant face recognition. A.I. Memo No. 1432, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[117] M. Dalla Serra and R. Brunelli. On the use of the Karhunen-Loeve expansion for face recognition. Technical Report 9206-04, I.R.S.T., 1992.

[118] M.A. Shackleton and W.J. Welsh. Classification of facial features for recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 573–579, Lahaina, Maui, Hawaii, 1991.

[119] A. Shashua. Correspondence and affine shape from two orthographic views: Motion and Recognition. A.I. Memo No. 1327, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1991.

[120] A. Shashua. *Geometry and Photometry in 3D visual recognition.* PhD thesis, M.I.T Artificial Intelligence Laboratory, AI-TR-1401, November 1992.

[121] Amnon Shashua. Algebraic functions for recognition. A.I. Memo No. 1452, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1994.

[122] Pawan Sinha. Object recognition via image invariances. *Investigative Ophthalmology and Visual Science*, 35(4):1626, 1994.

[123] John Stewman and Kevin Bowyer. Creating the perspective projection aspect graph of polyhedral objects. In *Proceedings of the International Conference on Computer Vision*, pages 494–500, Tampa, FL, December 1988.

[124] T.J. Stonham. Practical face recognition and verification with WISARD. In M. Jeeves, F. Newcombe, and A. Young, editors, *Aspects of Face Processing*, pages 426–441. Martinus Nijhoff Publishers, Dordrecht, 1986.

[125] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. In *Proceedings Image Understanding Workshop*, volume II, pages 843–850, Monterey, CA, November 1994.

[126] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[127] Demetri Terzopoulos and Keith Waters. Analysis of facial images using physical and anatomical models. In *Proceedings of the International Conference on Computer Vision*, pages 727–732, Osaka, Japan, December 1990.

[128] Sebastian Thrun and Tom M. Mitchell. Learning one more thing. Technical Report CMU-CS-94-184, Carnegie-Mellon University, 1994.

[129] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[130] Shimon Ullman and Ronen Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[131] Thomas Vetter, Anya Hurlbert, and Tomaso Poggio. View-based models of 3D object recognition: Invariance to imaging transformations. *Cerebral Cortex*, 3:261–269, May/June 1995.

[132] Thomas Vetter and Tomaso Poggio. Linear object classes and image synthesis from a single example image. A.I. Memo No. 1531, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1995.

[133] J.M. Vincent, J.B. Waite, and D.J. Myers. Location of feature points in images using neural networks. *BT Technology Journal*, 10(3):7–15, July 1992.

[134] Keith Waters and Demetri Terzopoulos. Modelling and animating faces using scanned data. *The Journal of Visualization and Computer Animation*, 2:123–128, 1991.

[135] John J. Weng, N. Ahuja, and T.S. Huang. Learning recognition and segmentation of 3-D objects from 2-D images. In *Proceedings of the International Conference on Computer Vision*, pages 121–128, Berlin, May 1993.

[136] Lance Williams. Performance-driven facial animation. In *SIGGRAPH '90 Proceedings*, pages 235–242, Dallas, TX, August 1990.

[137] Lance Williams. Living pictures. In *Models and Techniques in Computer Animation*, pages 2–12. Springer-Verlag, Tokyo, 1993.

[138] George Wolberg. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, California, 1990.

[139] K.H. Wong, Hudson H.M. Law, and P.W.M. Tsang. A system for recognizing human faces. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 1638–1642, 1989.

[140] Chyuan Jy Wu and Jun S. Huang. Human face profile recognition by computer. *Pattern Recognition*, 23(3/4):255–259, 1990.

[141] Rolf P. Würtz. *Multilayer Dynamic Link Networks for Establishing Image Point Correspondences and Visual Object Recognition.* PhD thesis, Bochum University, Germany, 1994.

[142] Yaser Yacoob and Larry Davis. Computing spatio-temporal representations of human faces. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 70–75, Seattle, WA, 1994.

[143] Malcolm P. Young and Shigeru Yamane. Sparse population coding of faces in the inferotemporal cortex. *Science*, 256:1327–1331, May 1992.

[144] Alan L. Yuille, Peter W. Hallinan, and David S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.