MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL INFORMATION PROCESSING
WHITAKER COLLEGE

A.I. Memo No. 1559 December 1995
C.B.I.P. Paper No. 128

Model-Based Matching of Line Drawings by Linear Combinations of Prototypes

Michael J. Jones and Tomaso Poggio

This paper also appeared in the Proceedings of the Fifth International Conference on Computer Vision

Abstract

We describe a technique for finding pixelwise correspondences between two images by using models of objects of the same class to guide the search. The object models are "learned" from example images (also called prototypes) of an object class. The models consist of a linear combination of prototypes. The flow fields giving pixelwise correspondences between a base prototype and each of the other prototypes must be given. A novel image of an object of the same class is matched to a model by minimizing an error between the novel image and the current guess for the closest model image. Currently, the algorithm applies to line drawings of objects. An extension to real grey level images is discussed.

This paper describes research done at the Artificial Intelligence Laboratory and within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences at the Massachusetts Institute of Technology. This research is sponsored by grants from ONR under contract N00014-93-1-0385 and from ARPA-ONR under contract N00014-92-J-1879; and by a grant from the National Science Foundation under contract ASC-9217041 (this award includes funds from ARPA provided under the HPCC program). Additional support is provided by the North Atlantic Treaty Organization, ATR Audio and Visual Perception Research Laboratories, Mitsubishi Electric Corporation, Sumitomo Metal Industries, Kodak, Daimler-Benz and Siemens AG. Support for the A.I. Laboratory's artificial intelligence research is provided by ARPA-ONR contract N00014-91-J-4038. Tomaso Poggio is supported by the Uncas and Helen Whitaker Chair at MIT's Whitaker College.

1 Introduction

The problem of image correspondence is basic to computer vision and arises in a number of vision applications such as stereo disparity, object recognition and motion estimation. General solutions such as optical flow techniques for computing the pixelwise correspondences between two images only work when the differences between the two images are relatively small. When the two images have large differences such as large rotations or changes in shape, then general methods for computing correspondences break down. For many applications, prior knowledge is available about the contents of the images for which the correspondence is being computed. This knowledge may be exploited in order to create a more robust correspondence algorithm. This is the approach discussed in this paper. We describe an algorithm for model-based matching which uses a simple model of a class of objects to find the correspondence between a novel view of an object of the same class and a standard "prototypical" view. Instead of using 3D models for objects, we build models from 2D example views of the objects. Our technique requires that the pixelwise correspondences between each example view and the standard example view be given by the user (presumably by semiautomatic techniques) in the training stage. Currently we are concerned with matching line drawings although straightforward extensions should allow the algorithm to be used with real images. Hence, this paper focuses on models of the shape of objects which do not take into account their textures.

2 Related work

Other researchers have studied techniques for constraining the search for correspondences by assuming a model for the form of valid flow fields. For example, Cootes and Taylor ([5, 6, 7]) proposed Active Shape Models (ASMs) which is similar to the approach we are taking. An ASM is built by first manually identifying a number of control points on a real image of an object. After the same control points are identified on a number of different images of the same object, a principal components analysis is done on the matrix consisting of vectors of control points. This yields a set of eigenvectors which describe the directions (in control point space) of greatest variation along which the control points change. An ASM is then the linear combination of eigenvectors plus parameters for translation, rotation and scaling. An ASM is matched to a novel image of the object by an algorithm that searches a region in the novel image around the current position of each control point to find a position of better fit for each control point and then updates the parameters of the ASM accordingly. Two of the main differences of their approach relative to ours are the fitting algorithm used (ours is a gradient based approach) and the use of a dense pixelwise flow field as opposed to a sparse vector of control points. Also, Cootes and Taylor match shape models (which are basically line drawings) to real images whereas we match line drawings to line drawings and also describe a method for matching real image models to real images.

Another group of researchers, Bergen, Anandan, Hanna and Hingorani [1], have described a framework for grey-level motion estimation. Their work is based on defining an error function which must be minimized to find the optimal flow field between two images. The error function they use is the sum of squared differences between one image and a warping of the other image according to the current estimate of the flow field. Bergen et al. constrain the flow field to adhere to some preselected form or model. The error is then minimized with respect to the parameters of the model by the Gauss-Newton minimization algorithm. The particular model used to constrain the flow can be selected according to the particular application. The ones discussed in Bergen et al. are rather general: affine flow, planar surface flow, rigid body motion and general optical flow. The main difference between their work and ours is the type of model used. Our models are learned from examples and are specific to a particular object class.

The main motivation for our work is the linear class concept of Poggio and Vetter [11, 9] that justifies modeling an object in terms of a linear combination of prototypes. Poggio and Vetter showed that linear transformations can be learned exactly from a small set of examples in the case of linear object classes. Furthermore, many object transformations such as 3D rotations of a rigid object and changing expression of a face can be approximated by linear transformations, that can be learned from a small number of examples. The same motivation underlies the work of Beymer [2] who describes an alternative approach, also based on a linear combination of prototypes, to vectorize grey-level images.

3 Model-based matching using prototypes

3.1 The model

We would like the models used for model-based matching to be learned from examples as opposed to being hardwired. To learn a model, a number of examples or prototypes of an object are given which show how the object can change. For example, to learn a model of a face with varying pose and facial expression, several examples of the face at different poses and with different expressions would be given to the system.

In addition to the prototype images, we require that pixelwise correspondences be given between one of the prototypes (usually the "average" prototype) which is chosen to be the base image and each of the other prototypes. In practice the correspondences are specified by the user during this "learning" stage in a semiautomatic way using special tools.

Given the correspondences, each prototype can be "vectorized" - written as a vector of points. In practice each prototype is represented as two matrices, one with the displacements in the x direction from each point in the base image to the corresponding point in the prototype and one with the y displacements. We define a model in this framework to be a linear combination of vectorized prototypes or equivalently a linear combination of example flow fields (see also [10, 3]).

To write the models mathematically, we must first introduce some notation. Let I_0 be the base prototype image to which all the correspondences reference. Let N be the total number of prototypes. Let $\boldsymbol{Dx}_i$ be the matrix of displacements in the x direction mapping the coordinates of base image I_0 to the corresponding coordinates of prototype I_i. Similarly, let $\boldsymbol{Dy}_i$ be the matrix of y displacements. Together, $\boldsymbol{Dx}_i$ and $\boldsymbol{Dy}_i$ make up a flow field. The model images consist of all images whose flow field is a linear combination of the prototype flow fields plus an affine transformation. In symbols,

$$\boldsymbol{Dx}' = \sum_{i}^{N-1}(c_i \boldsymbol{Dx}_i) + p_0\boldsymbol{X} + p_1\boldsymbol{Y} + p_2$$

$$\boldsymbol{Dy}' = \sum_{i}^{N-1}(c_i \boldsymbol{Dy}_i) + p_3\boldsymbol{X} + p_4\boldsymbol{Y} + p_5$$

The $\boldsymbol{Dx}'$ and $\boldsymbol{Dy}'$ matrices are the flow field describing model image I'. Each row of the constant matrix $\boldsymbol{X}$ is $(-w/2, -w/2+1, ..., -1, 0, 1, ..., w/2-1, w/2)$ where w is the width of the prototype images. Similarly, each column of the constant matrix $\boldsymbol{y}$ is $(-h/2, -h/2+1, ..., -1, 0, 1, ..., h/2-1, h/2)^T$ where h is the height of the images.

These equations describe the flow fields for the model images. To actually get the grey level representation of I', it is necessary to warp base image I_0 according to $\boldsymbol{Dx}'$ and $\boldsymbol{Dy}'$ and thereby *render* the matrices $\boldsymbol{Dx}'$ and $\boldsymbol{Dy}'$ as a black and white image. If the warp function simply moves pixels in the base image according to the flow field (without doing any blurring or hole filling) then a model image can be written

$$I'(x + \boldsymbol{Dx}'(x,y), y + \boldsymbol{Dy}'(x,y)) = I_0(x,y).$$

To obtain prototype line drawings and the associated correspondences in practice, a drawing program is used. A model of a new object is made by first creating a line drawing of the base image. The base image is usually the approximate average image in terms of the various object transformations one wants to represent. Next, new examples of the object are drawn by changing the lines and curves of the base prototype. The pixelwise correspondences between the base prototype and each additional prototype can then be computed automatically since the equations describing the lines and curves in each prototype are known. A typical example base of prototype images is shown in figure 1.

3.2 Matching novel images

Now that the prototypes have been defined, we want to use them to find the pixelwise correspondence between the base prototype and a novel image that is in the same object class as the prototypes. The general strategy for matching the novel image will be to define an error between the novel image and the current guess for the closest model image after rendering it and then try to minimize this error with respect to the linear coefficients c_i and the affine parameters p_i. Following this strategy, we define the sum of squared differences error

$$E(\boldsymbol{c},\boldsymbol{p}) = \frac{1}{2}\sum_{x,y}[I^{novel}(\hat{x},\hat{y}) - I^{model}(\hat{x},\hat{y})]^2$$

where

$$\hat{x} = x + \sum_{i=1}^{N-1} c_i \boldsymbol{Dx}_i(x,y) + p_0 x + p_1 y + p_2,$$

$$\hat{y} = y + \sum_{i=1}^{N-1} c_i \boldsymbol{Dy}_i(x,y) + p_3 x + p_4 y + p_5,$$

the sum is over all pixels (x,y) in the images, I^{novel} is the novel grey level image being matched and I^{model} is the model grey level image. Assuming the simplest warping function,

$$I^{model}(\hat{x},\hat{y}) = I_0(x,y).$$

In this case, the error can be written

$$E(\boldsymbol{c},\boldsymbol{p}) = \frac{1}{2}\sum_{x,y}[I^{novel}(\hat{x},\hat{y}) - I_0(x,y)]^2$$

The sum of squared differences error depends on the model parameters and gives a measure of the distance between the novel image and the current guess for the model image. Minimizing the error yields the model image which best fits the novel image.

In order to minimize the error function, the Levenberg-Marquardt algorithm ([12]) is used (a similar use of Levenberg-Marquardt is described in [13]). This algorithm requires the derivative of the error with respect to each parameter. The necessary derivatives are as follows:

$$\frac{\partial E}{\partial c_i} = \sum_{x,y}[(I^{novel}(\hat{x},\hat{y}) - I_0(x,y))\ \frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial c_i}]$$

$$\frac{\partial E}{\partial p_i} = \sum_{x,y}[(I^{novel}(\hat{x},\hat{y}) - I_0(x,y))\ \frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial p_i}]$$

$$\frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial c_i} = \frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial c_i} + \frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial c_i}$$

$$\frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial p_i} = \frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial p_i} + \frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial p_i}$$

$$\frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial \hat{x}} \approx \frac{I^{novel}(\hat{x}+1,\hat{y}) - I^{novel}(\hat{x}-1,\hat{y})}{2}$$

$$\frac{\partial I^{novel}(\hat{x},\hat{y})}{\partial \hat{y}} \approx \frac{I^{novel}(\hat{x},\hat{y}+1) - I^{novel}(\hat{x},\hat{y}-1)}{2}$$

$$\frac{\partial \hat{x}}{\partial c_i} = \boldsymbol{Dx}_i(x,y)$$

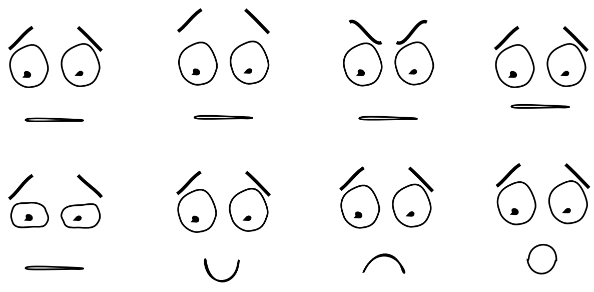

Figure 1: A typical example base of prototype line drawings.

$$\frac{\partial \hat{y}}{\partial c_i} = \boldsymbol{Dy}_i(x,y)$$

$$\frac{\partial \hat{x}}{\partial p_0} = x, \quad \frac{\partial \hat{x}}{\partial p_1} = y, \quad \frac{\partial \hat{x}}{\partial p_2} = 1,$$

$$\frac{\partial \hat{x}}{\partial p_3} = 0, \quad \frac{\partial \hat{x}}{\partial p_4} = 0, \quad \frac{\partial \hat{x}}{\partial p_5} = 0$$

$$\frac{\partial \hat{y}}{\partial p_0} = 0, \quad \frac{\partial \hat{y}}{\partial p_1} = 0, \quad \frac{\partial \hat{y}}{\partial p_2} = 0,$$

$$\frac{\partial \hat{y}}{\partial p_3} = x, \quad \frac{\partial \hat{y}}{\partial p_4} = y, \quad \frac{\partial \hat{y}}{\partial p_5} = 1$$

Given these derivatives, the Levenberg-Marquardt algorithm can be used straightforwardly to find the optimal $\boldsymbol{c}$ and $\boldsymbol{p}$. Notice that the algorithm is a sequence of *vectorization* and *rendering* (through warping) steps.

3.3 Improving performance

Implementing the minimization described in the previous section using line drawings as prototypes does not work well when the initial model parameters are far from the optimal ones. There are a couple of standard techniques we can use that improve the performance of the matching significantly.

The first improvement is to simply blur the line drawings. Since only the black pixels are important in a line drawing, a blurring algorithm is used which only blurs the black pixels onto the white background. Using blurred line drawings makes the minimization more robust in the sense that the initial parameters can be much further away from the optimal ones for the minimization to succeed.

The second improvement is to use a coarse-to-fine approach. This is a standard technique in computer vision ([4]). The idea is to create a pyramid of images with each higher level of the pyramid containing an image that is one fourth the size of the one below. The flow fields must also be subsampled, and all x and y displacements must be divided by 2. Levenberg-Marquardt is used to fit the model parameters starting at the coarsest level, and then these parameters are used as the starting point at the next level. The constant affine parameters (p_2 and p_5) must be multiplied by 2 as they are passed down the pyramid to account for the increased size of the images.

The coarse-to-fine approach also significantly improves the robustness of the matching. When combined with blurring, the matching algorithm works well for a large range of settings of the initial parameters.

A stochastic gradient minimization algorithm (described in [14]) has also been tried in place of Levenberg-Marquardt. It was found to be much faster (around 25 times) and more robust in that it got caught in local minima less frequently. The results reported here are with the Levenberg-Marquardt algorithm because the stochastic gradient algorithm was implemented after the first draft of this paper.

3.4 Pseudo code

The following pseudo code describes the matching algorithm.

1. Load novel image, I^{novel}
2. Load base prototype, I_0, and flow fields for the other prototypes, $\boldsymbol{Dx}_i$ and $\boldsymbol{Dy}_i$
3. Create image pyramids for I^{novel} and I_0 and for each $\boldsymbol{Dx}_i$ and $\boldsymbol{Dy}_i$
4. Blur all images in novel image pyramid
5. Initialize parameters $\boldsymbol{c}$ and $\boldsymbol{p}$ (typically set to zero)

For each level in the pyramid

6. Estimate the parameters $\boldsymbol{c}$ and $\boldsymbol{p}$ using Levenberg-Marquardt

   When computing the error in Levenberg-Marquardt, the model image is created by warping I_0 according to the current linear combination of prototype flow fields plus affine parameters and then the resulting model image is blurred.

7. Multiply the constant affine parameters p_2 and p_5 by 2
8. Go to next level

9. Output the parameters

3.5 Results

Some preliminary tests have been done using our approach to model-based matching. In one such test, the prototype images in figure 1 were used to create

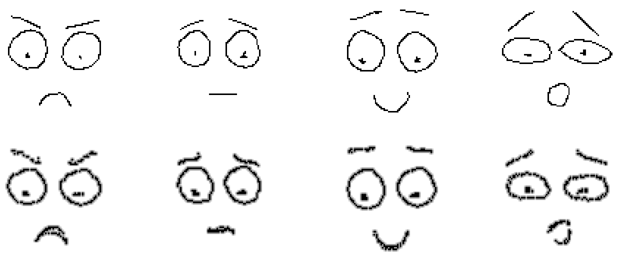

Figure 2: Results of matching novel images using the prototypes in figure 1 . The novel images are in the top row and the model images which were estimated are in the bottom row.

a model of simple cartoon faces. The pixelwise correspondences between each prototype and the base prototype were obtained using the output of the drawing program on which the images were generated. The base prototype is the face in the upper left corner of figure 1.

Novel images which were similar to those in the example base were created by hand. These images were drawn so that they were roughly normalized for translation, scale and rotation. Figure 2 shows the results of fitting the model to the novel images. The top row of images are the novel images and the bottom row are the closest model images as estimated by the matching algorithm described above. The model parameters were all initialized to zero, which means the base prototype was used as the starting point for the matching algorithm. As the figure shows, the algorithm did a good job of finding a model image which matched well with each novel image. The lines in the model images are thicker due to a small amount of blurring that is done after warping in order to fill in "holes" left by warping. All model images are generated from their respective flow fields by warping the base image.

4 Extensions

4.1 A general hierarchical componentwise approach

An affine transformation is included in the model because it allows for the novel image being matched to have moderate changes in scale, rotation and translation from the model prototypes. In other words, the affine parameters provide some extra tolerance in the model. Of course, the affine parameters are global in the sense that they scale the whole image or rotate the whole image as opposed to affecting only a piece or a single feature of the image. This fact exposes one of the problems with the approach just described. It is brittle to translations, rotations or scaling of only a single feature in the image if this local variation is not accounted for by some of the prototypes. This is more of a problem for matching novel line drawings that a user has complete freedom in creating than with real images which are constrained by the physical world.

One obvious solution to this problem is to use a componentwise approach in which images are treated as being composed of several different components, say eyes, mouth and nose. Each component would have its own model using the same formulation as in the previous section. In other words, each component is specified by a number of prototypes along with the pixelwise correspondences for each prototype. These components are then combined to form a complete image, say of a full face, by specifying where each component can be located. The location information is again specified using a number of prototypes for the whole image. These image prototypes would simply consist of x, y locations for each component. A number of image prototypes would be needed to show how each component could change location relative to the other components. The new componentwise model would be a linear combination of location vectors as well as a linear combination of individual component prototypes.

We are extending this componentwise idea towards a potentially powerful hierarchical framework to allow more complicated images (with possibly multiple objects). The idea is to build components from a linear combination of component prototypes and then build simple objects from a linear combination of positions of components and then build more complicated objects from a linear combination of positions of simple objects and so on.

4.2 Using real images

Another ongoing extension to this work is to apply the matching algorithm to real grey level and color images as opposed to black and white line drawings.

In this case, in addition to modeling the shape of objects, we also model the texture of objects. We model texture analogously to the way we modeled shape - as a linear combination of the grey level values (texture) of the prototype images (see also [2], for an alternative approach to the same problem). A rather general justification of models of shape and texture consisting of linear combinations of prototypical shapes and textures is the following. Under weak assumptions, one can prove that if any network can learn to synthesize shape or texture from examples then the desired shape or texture must be well approximated by a linear combination of the examples (see [3, 8]).

Let $\{I_j\}$ be the set of prototype images where I_0

is the base image. Define DI_j, the image of intensity differences between I_j and I_0, as

$$DI_j(x,y) = I_j(x+\boldsymbol{Dx}_j(x,y), y+\boldsymbol{Dy}_j(x,y)) - I_0(x,y).$$

For any (x,y) in the base image, the corresponding model point is

$$I^{model}(\hat{x},\hat{y}) = I_0(x,y) + \sum_{j=1}^{N-1} b_j DI_j(x,y)$$

where

$$\hat{x} = x + \sum_{i=1}^{N-1} c_i \boldsymbol{Dx}_i(x,y) + p_0 x + p_1 y + p_2$$

$$\hat{y} = y + \sum_{i=1}^{N-1} c_i \boldsymbol{Dy}_i(x,y) + p_3 x + p_4 y + p_5.$$

In other words, the new position of the pixel at location (x,y) in the base image is determined by a linear combination of prototype positions (given by $\boldsymbol{Dx}_i(x,y)$ and $\boldsymbol{Dy}_i(x,y)$), and the new grey level value of the pixel is determined by a linear combination of prototype grey level values for that pixel. The two linear combinations, for shape and texture respectively, use the same set of prototype images but two different sets of coefficients.

To match a novel grey level image, we can still use Levenberg-Marquardt. The minimization is now with respect to the vector of grey level coefficients $\boldsymbol{b}$ as well as to $\boldsymbol{c}$ and $\boldsymbol{p}$.

5 Applications

5.1 Image analysis

One problem that model-based matching can be applied to is the problem of image analysis. By image analysis we mean the problem of determining certain parameters describing an image such as the pose or expression parameters of an image of a face for example. Our approach to image analysis is to learn a mapping from images to their corresponding parameters (see [3]). The representation used for the images is critical in this approach. For example, trying to find a mapping from the raw grey level matrix of an image to its associated parameters would not result in a mapping which generalized to new images. This is because the grey level values of an image do not change smoothly as the objects in the image change smoothly. Instead of using the grey level representation, Beymer et al. find the pixelwise correspondences for each example image and use the vector of labelled points for each image as the image representation. They call the vector of labelled points the "vectorized" representation of an image. Thus to analyze a new image, it must first be converted into the vectorized representation. To do this we can use the model-based matching approach previously described instead of other techniques such as optical flow. Thus, our approach to image analysis is to first define a model as described in section 3.1 from a set of prototype images and their flow fields. The analysis parameters (such as pose) are also given for each prototype. A mapping is then learned which maps the vectorized prototypes to their corresponding analysis parameters. A novel image is analyzed by first matching the linear combination of prototypes model to the image as described in section 3.2. After matching, the resulting correspondences are used to create the vectorized representation for the novel image. The parameters of the novel image are then calculated by applying the previously learned mapping to the vectorized representation of the novel image as described in [3].

As described briefly in section 3.5, we have written a system for analyzing line drawings such as those in figure 1. The system learns to analyze sketches from a user who trains the system with prototype examples. The system is first trained with prototypes of line drawings of an object along with the pixelwise correspondences. Given a set of prototypes, the system attempts to match a novel line drawing which is approximately in the space of images spanned by the prototypes using the algorithm of section 3.4. The model parameters which are found by the matching can be used as the analysis parameters for the image. Alternatively, the model parameters can be mapped by an approximation network to a possibly higher level set of analysis parameters (see [3]. Examples of the higher level parameters would be given with each prototype.

5.2 Man-machine interface

Image analysis can be used to build a general man-machine interface or a gesture recognition system ([3]). For example, if a model of a hand were built from example views of a hand then novel views of a hand could be analyzed to recover their position and orientation. These parameters could then be used as input to a computer to control things the same way a 3D mouse does. Other possibilities for a man-machine interface are analyzing facial expression and using it as input to the computer.

Other potential applications for model-based matching are object recognition, very low bandwidth teleconferencing and virtual reality simulations.

6 Discussion and conclusions

We have described a robust algorithm for model-based matching. Using object models to guide the matching algorithm may be essential in cases where the differences between two images of an object are too great for a general correspondence algorithm to work well. The need for prior knowledge in the form of object models comes from the fact that optical flow is an underconstrained problem although other ways of adding constraints have of course been used (see for example [1, 13]).

The linear combination of prototypes model that we described has several advantages. It is a simple learning-from-examples model that only requires 2D views as opposed to a 3D model. It has a quite deep motivation since the linear combination of prototypes model is intimately related to general properties of a very broad class of synthesis networks of the type described by [3]. A new model is fairly simple to create since all that is required are a number of example views of the object class and the pixelwise correspondences for each. Most importantly, the matching algorithm works well in practice. One problem with this approach is the need for the correspondences for each prototype. In general we expect that once a good *vocabulary* of models is created, new models will not need to be created very often.

Acknowledgements: The authors would like to thank Paul Viola for suggesting the stochastic gradient descent algorithm and for help in implementing and testing it.

References

[1] James R. Bergen, P. Anandan, Keith J. Hanna and Rajesh Hingorani. Hierarchical Model-Based Motion Estimation. In *Second European Conference on Computer Vision*, Springer-Verlag, Santa Margherita Liguere, Italy, May 1992, pp. 237-252.

[2] David Beymer. Vectorizing Face Images by Interleaving Shape and Texture Computations. To be published as AI Memo, AI Laboratory, MIT 1995.

[3] D. Beymer, A. Shashua and T. Poggio. Example Based Image Analysis and Synthesis. AI Memo No. 1431, AI Laboratory, MIT 1993.

[4] Peter J. Burt. The Pyramid as a Structure for Efficient Computation. in *Multi-Resolution Image Processing and Analysis.* ed. Rosenfield. Springer-Verlag, 1984, pp. 6-37.

[5] T.F. Cootes and C.J. Taylor. Active Shape Models - "Smart Snakes". In *Proceedings of the British Machine Vision Conference.* Springer-Verlag, 1992, pp. 266-275.

[6] T.F. Cootes, C.J. Taylor, D.H. Cooper and J. Graham. Training Models of Shape from Sets of Examples. In *Proc. of the British Machine Vision Conference.* Springer Verlag, 1992, pp. 9-18.

[7] T.F. Cootes and C.J. Taylor. Using Grey-Level Models to Improve Active Shape Model Search. In *International Conference on Pattern Recognition*, vol 1. IEEE Computer Society Press, 1994, pp 63-67.

[8] Federico Girosi, Michael Jones and Tomaso Poggio. Priors, Stabilizers and Basis Functions: from regularization to radial, tensor and additive splines. AI Memo No. 1430, AI Lab, MIT 1993.

[9] Nikos K. Logothetis, Thomas Vetter, Anya Hurlbert and Tomaso Poggio. View-based Models of 3D Object Recognition and Class-specific Invariances. AI Memo No. 1472, AI Lab, MIT 1992.

[10] Tomaso Poggio and Roberto Brunelli. A Novel Approach to Graphics. AI Memo No. 1354, AI Laboratory, MIT 1992.

[11] Tomaso Poggio and Thomas Vetter. Recognition and Structure from one 2D Model View: Observations on Prototypes, Object Classes and Symmetries. AI Memo No. 1347, AI Laboratory, MIT 1992.

[12] 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, second edition, 1992.

[13] Richard Szeliski and James Coughlan. *Spline-Based Image Registration.* Technical Report 94/1, Digital Equipment Corporation, Cambridge Research Lab, April 1994.

[14] Paul A. Viola. *Alignment by Maximization of Mutual Information* AI Technical Report 1548, AI Laboratory, MIT 1995.