Example-Based Analysis and Synthesis for Images of Human Faces

by

Tony F. Ezzat

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Electrical Engineering and Computer Science
and
Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1996

© Massachusetts Institute of Technology 1996

Signature of Author .....
Department of Electrical Engineering and Computer Science
October 2, 1999

Certified by .....
Tomaso Poggio
Uncas and Helen Whitaker Professor, Department of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by .....
Frederic Morgenthaler
Chairman, Departmental Committee on Graduate Students

Example-Based Analysis and Synthesis for Images of Human Faces

by

Tony F. Ezzat

Submitted to the Department of Electrical Engineering and Computer Science
on October 2, 1999, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Electrical Engineering and Computer Science
and
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

We describe image-based synthesis techniques that make possible the creation of computer models of real human faces. The computer model is built using example images of the face, bypassing the need for any three-dimensional models. Intermediate views are synthesized through the use of a trained learning network which associates each of the example images with a set of pose and expression parameters. Control of the model is achieved by using the trained network to synthesize a new image of the face for any desired setting of pose and expression parameters. Specifically, we describe the creation of

- a model that synthesizes vertical and horizontal pose movements of the head
- a model that synthesizes various mouth expressions, and
- a model that synthesizes pose movements, eye movements, and mouth movements combined.

We will also describe analysis techniques to "analyze" novel image streams of the same human face using the trained synthesis networks. For each incoming novel frame, these analysis techniques estimate the set of parameters that best match the model. The analysis algorithms are robust to translation, onplane rotation, scale, lighting changes, background changes, and even radical hairstyle changes!

Together, analysis and synthesis networks can serve as a basis for model-based compression that may be useful for video email and video teleconferencing. Other applications include human-driven animation(analysis alone) and synthetic human characters and avatars (synthesis alone).

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

Acknowledgments

Firstly, I would like to thank my thesis advisor, Tomaso Poggio, for the strong support, encouragement, and advice that he has given me since I have come to the MIT AI Lab. Without his guidance and faith, this thesis would not be possible.

David Beymer, Mike Jones, Steve Lines, and Federico Girosi were instrumental for this work. All four provided crucial advice and discussion on various aspects of this thesis. David's work in [5] and Mike's work in [14] served as bases for this thesis work, and the reader is encouraged to take a look at those papers before reading this thesis.

I would also like to thank Kah-Kay Sung and Sajit Rao, for numerous useful discussions, as well as for making the AI lab and CBCL an exciting and enjoyable environment to work in.

It is also important to acknowledge David Sarnoff Research Labs for allowing the use of their optical flow code and image libraries.

Finally, I am grateful to my parents, my brother, and my grandparents for their constant encouragement, support, and patience. This thesis is dedicated to them.

Is it the Search that soothes,
or the Discovery that shakes our foundation?
—Sufi poet

Contents

Chapter 1

Introduction

1.1 Overview

The term *model-based coding* denotes a scheme of the kind shown in figure 1-1. A video sequence containing one or more moving objects is *analyzed* using computer vision techniques to yield information about the size, location, and motion of the objects. Typically, this information is extracted from the sequence using *models* of each object (hence the origin of the name model-based coding.) The parameters needed to animate the model may then be transmitted to the receiver, which *synthesizes* the sequence.

Interest in model-based coding is widespread at the present time, and general reviews of the current literature may be found in [2] [21]. This interest arises from the potentially large reductions in the bit rate that model-based coding promises over the current generation of hybrid interframe coders, represented by the H.261, H.263, MPEG-1, and MPEG-2 standards. The larger reductions in the bit rate are achieved by model-based techniques because the parameters extracted encode *high-level* attributes of objects, while the current standards can only perform statistical decorrelation, which makes no use of any prior knowledge about the particular object being coded.

Most of the experimental work in model-based coding has been concerned with modelling and coding human heads. This is because of possible applications in videotelephony and videoconferencing, where it is desirable to reduce the bitrates to allow for phone-line transmissions. The head is in some respects an easy object to model in these applications, for there is usually not much lateral movement and not much rotation. On the other hand, the human face has a flexible rather than a rigid shape, with a complex set of controlling muscles; this makes accurate analysis and synthesis fairly difficult. This thesis work falls into this category, and will focus on new model-based techniques for the analysis and synthesis solely of images of human faces (hence the title).

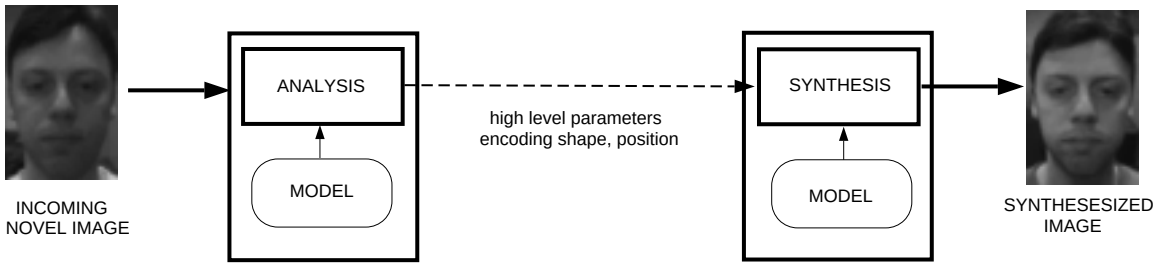


Figure 1-1: General diagram of model-based coding schemes.

1.2 Background

1.2.1 Graphics-Based Models

The most important aspect of model-based coding, as has been pointed out in the literature [2], is the particular facial model that is adopted for analysis and synthesis purposes. The reasons for this are numerous. For example, the particular parametrization of the model defines the set of attributes that the analysis algorithms can extract from the novel images. Also, the nature of the model affects the ease with which it can be compared to the novel incoming image, and the degree to which it can realistically synthesize the facial motion at the synthesis end. The manner in which the model is constructed determines how easy it is to build models for other objects. Finally, the nature of the model plays a big role in the time complexity of the analysis and synthesis algorithms.

Many of the early attempts at facial model-based coding involved modelling the human face in three dimensions using computer graphics techniques. A lot of this graphics-based work is based on that of Parke's [20], who in the 70's developed parametrized models of the human face as a tool for computer-assisted animation. Many researchers have augmented the wireframe model with more realistic facial models to be able to capture more complex nuances of the face. For example, Terzopoulos and Waters [24] incorporated a physics-based synthetic tissue model into their facial model to simulate the properties of human facial skin. The model is composed of an assembly of several layers of point masses connected by springs. The springs are assigned different stiffnesses in accordance with the inhomogeity of real facial tissue. In addition to a skin model, Terzopoulos and Waters also incorporated muscle actuators to mimic the muscle fibers that manipulate skin tissue.

Ekman and Friesen's work, [9], provided a necessary level of control for many of these graphics-based models: their facial action coding system (FACS) quantified facial expressions in terms of 44 action units (AU's) involving one or more muscles and associated activation levels. The importance of Ekman and Fiesen's work is that it allows one to try to map the extracted information from the incoming image sequences into the action units that control the graphic model, and many researchers, [1] [24] [10], have tried to use a subset of the action units to manulate a graphics

model at the synthesis end.

Texture-mapping techniques, borrowed from the computer graphics literature, have also come to play an important role in making the computer models look more realistic [1], [27] [29]. A facial image of the talker in the sequence is extracted and projected onto the graphics model. To be able to handle mouth and eye deformations in addition to pose and lateral head movements, a *clip-and-paste* approach is usually adopted, where the eye and mouth regions from the original sequence are extracted and blended onto the final model at the synthesis end.

1.2.2 Image-Based Rendering

More recently, however, a new synthesis approach has begun to emerge, whose basis is to completely forego any underlying computer graphics models, and instead adopt an *image-based model*. The underlying motivation is that it is very hard to achieve reasonable degrees of synthetic realism using models based on computer graphic techniques because it is very hard to model facial muscular and skin tissue. In addition, as the modelling complexity increases to improve the realism, the rendering latency also increases. The philosophy behind the new image-based rendering approaches is to attempt to ease some of these modelling and rendering problems by using a set of images to model objects. In doing so, not only is it hoped that there will be no need to model facial muscular and skin tissue, but also that the rendering time will remain roughly constant no matter how complex the imagery becomes.

Image-based rendering techniques probably have their origin in the work of Lippman [18], who put together an image-based display system called Movie-Maps that allowed one to virtually and interactively explore a city. The system would show the user pre-recorded sequences of a car traveling through a street, and, at each intersection, the user had the option to choose which direction he wanted to turn. Based on the user's choice, the system would load the correct turn sequence, and, following that, the correct new street sequence. As Lippman himself discusses in the paper, the system is not capable of generalizing beyond the set of sequences, viewpoints, and frame rates which the system was recorded at.

Subsequent attempts at the creation of image-based rendering systems have tried to extract photometric information from the set of images to be able to generalize, in some way, from the given views [17] [19] [16] [7]. A particularly important and relevant attempt in this regard is the work of Chen and Williams [8], who showed that a *morphing* technique was suitable for the generation of realistic, novel, intermediate images, given a set of image endpoints. Image morphing, introduced in [3], is the simultaneous interpolation of shape and texture. The technique generally involves two steps: The first step establishes correspondence between two images and is the most difficult part of the morphing methods. The correspondence is usually first established manually for a small set of points, and is then automatically expanded to include the entire image. The second step in the process is to use the mapping to interpolate the *pixel positions* from the image endpoints, followed by *blending* the pixel values themselves. In their work, Chen and Williams used computer models to generate synthetic imagery for use in their morphing technique, and they obtained their

correspondence fields from the z-values of the the computer model itself. They were able to show that, by augmenting the morphing technique with hole-filling algorithms and view-independent depth priorities, the intermediate images were realistic.

Beymer, Shashua, and Poggio [5], upon which much of this thesis work is based, used precisely such a morphing approach at the synthesis end, and, furthermore, applied the morphing algorithms particularly to images of faces. Since there was no apriori depth information, the correspondence was obtained through the use of hierarchical, gradient-based optical flow algorithms [13], [4]. In specific, Beymer, Shashua, and Poggio showed that morphing based on optical flow correspondences was suitable for capturing pose changes and mouth movements such as smiles and frowns, as well as other types of morphs such as interpersonal morphs. Of course, due to the limitations in the flow algorithms, the images had to be similar and not far apart, or else the flow algorithms would fail. Nevertheless, the use of optical flow algorithms to obtain correspondence and generate realitic intermediate images is significant, given the fact that the algorithms obtain correspondence *automatically* – that is, no manual specification is needed.

1.2.3 Learning Networks for Synthesis

Another important contribution made in Beymer, Shashua, and Poggio [5] is the incorporation into the synthesis module of a *learning network framework* that was first introduced in Poggio and Brunelli [22]. The synthesis modules built by Beymer, Shashua, and Poggio associate each of the example images with a position in an imposed, multi-dimensional parameter space. A radial basis function [11], is then used to *learn the mapping from the parameter space to the space of correspondence vectors*. For incoming novel parameters, the radial basis function synthesizes a new correspondence vector that lies at that position in the network. The morphing techniques are subsequently employed to render the novel image using the synthesized correpondence vector and the example images in the network.

The importance of the adoption of this learning network framework should be clear: the learning framework allows one to easily *parametrize* the space of facial images. In the context of model-based coding, parametrization defines the set of high-level parameters that an analysis module should try to extract from the incoming image sequence, and send to the synthesis module for reconstruction. The set of chosen parameters, as shown in Beymer, Shashua, Poggio, may encode such things as degree of horizontal pose, degree of smile, degree of frown, etc. Furthermore, the radial basis function framework can map from a *multi-dimensional* parameter input space to a *multidimensional* correspondence vector space, which is useful since it could potentially allow for many different synthesis configurations.

1.2.4 Problems with Feature-based Normalization

Beymer, Shashua, Poggio also used a learning network framework to perform analysis. They used a radial basis function to learn the inverse mapping, *from the correspondence vectors to the set of parameters*. For any, new incoming novel image, optical

flow is used to obtain correspondence between the incoming image and the reference example in the set of example images. The radial basis function is then used to map from the obtained flow to a set of parameters. To be able to analyze novel facial images despite changes in scale, rotation, and position, Beymer, Shashua, Poggio also first normalized the incoming images by finding a set of eye features, setting the interocular axis to the horizontal position, and fixing the interocular distance to a predetermined length. The same normalization procedure was also performed on the example images, and the correspondences were all obtained with respect to these normalized examples and novel images.

Such a feature-based normalization scheme, however, had many associated problems, some of which included:

- Normalizing images by finding the eyes might not work in cases of images where the eyes are closed, and also leads to an unstable set of features in the case of eye movement. Experiments were performed in which the set of features used for normalization were the corners around the eye, but these features proved to be unstable due to the associated lack discriminating texture.

- Normalization by fixing the interocular axis and distance is very sensitive to slight movements in the pixel positions of the feature labels being used. If a set of feature labels were slightly closer to each other by a few pixels than they should have been (due to inaccuracy in the feature finder), this would cause the normalized image to be much larger than the corresponding reference image from the example set. This, in turn, led to an incorrect flow-field, which led to incorrect analysis.

- The eye labels were also the foundation for a set of nose and mouth labels which allowed the extraction of rectangular regions of flow around the nose and mouth regions for the purposes of regional analysis. However, whenever there was noise in the eye labels, there would also be noise in the corresponding nose and mouth labels, and thus, noise in the analysis as well.

- It was also noticed that the interocular distance, besides getting larger and smaller with scale changes in the image, also got larger and smaller with changes in pose. Consequently, fixing the interocular distance to a predetermined amount had the deleterious effect of making the images of a head looking left or right larger or smaller, which, in turn, led to incorrect flowfield analysis.

Essentially, all the problems described above involved the fact that inaccuracies in feature localization led to significant distortions in the correspondence fields used for analysis.

1.2.5 Analysis-by-Synthesis: Searching Across the Parameters

To alleviate the problems associated with a feature-based normalization approach, a different approach was taken in this thesis where the synthesis model itself is addi-

tionally parametrized with a set of affine parameters, besides the intrinsic synthesis parameters. Consequently, besides changing the pose of the head or the mouth expression, the head may also be translated, rotated in the plane, and scaled up or down. The augmentation of a network with these additional parameters now allows for an analysis algorithm that completely foregoes any prior feature-based normalization of the incoming novel image. Instead, the algorithm now *searches across the space of affine and intrinsic parameters* encoded by the synthesis network to find the set of parameters that best match the image-based model to the new, incoming image.

This type of algorithm, which involves *analysis-by-synthesis*, has been described before in the context of object recognition [25] and in the context of model-based coding [12]. Important and relevant work in this regard was made by Jones and Poggio [14], who constructed parametrized models of line drawings, and then used the Levenberg-Marquardt [23] algorithm to match the models to novel line drawings input by the user. The models themselves consisted of a linear combination of prototypes (as in [25]) which were placed into correspondence using a mixture of optical flow algorithms and manual techniques. The error metric which the Levenberg-Marquardt algorithm tried to minimize was the error between the novel drawing and the current guess for the closest model image. At every iteration, the algorithm would compute the gradient of this error metric with respect to the model parameters, and proceed to a new guess for a set of parameters that would produce a new model image closer to the novel image.

Another very similar analysis-by-synthesis approach was made by Kuch and Huang [15], who constructed a computer graphics model of a hand, and used it to analyze novel hand images. In this case, as well, an image-based metric was defined and minimized: at every iteration, the graphics model of the hand was rendered in binary and exclusively-OR'ed with a similarly binarized version of the novel image, to yield a residual error image. The number of high pixels in the residual error image is then summed to give a single scalar error value, which denotes the amount of mismatched pixels between the images. Instead of computing the gradient of this error with respect the hand model parameters, as in [14], Kuch and Huang instead *actually perturbed each of the hand model parameters locally and independently*, and chose to proceed to a new guess based on the perturbed parameters that reduced the overall error metric.

In this thesis, an *iterative, independent, and local paremeter perturbation* strategy similar to Kuch and Huang's was adopted to match the facial synthesis networks with the novel incoming facial sequences. Instead of minimizing an *image-based* error metric, however, as in [14] and [15], a *correspondence-based* metric was chosen for this thesis: at every iteration, the analysis-by-synthesis strategy attempts to reconstruct a *flow* from the synthesis network that matches a novel, incoming *flow* between two consecutive images in the novel sequence. In choosing to match flows rather than images, it is hoped that the analysis-by-synthesis algorithm will be less sensitive to local minima, and, in addition, be able to analyze images regardless of lighting changes. It should be noted that a flow-based metric for analysis was also essentially used in Beymer, Shashua, Poggio, but in a different manner.

Besides suffering from local minima, another drawback of analysis-by-synthesis schemes that need to search across a set of parameters to match the model to the

image, is that they may be prohibitively slow. In this regard, feature-based schemes involving finding the eyes or the head prior to performing model-matching may be needed to place the model in close proximity to the head in the novel image, and thus reduce the time needed for the analysis strategy to achieve a comfortable minimum. Note, however, that the manner in which such feature-based schemes are used should not alter the novel image (and thus, the flow fields) in any way. In fact, their use in this manner allows for errors in the feature localization, since the subsequent analysis procedure should itself be robust to moderate changes in rotations, scale, and translation.

1.3 Goal and Layout of the Thesis

On the synthesis side, the goal of this thesis was to build on the work done in Beymer, Shashua, Poggio [5] by exploring the degree to which the synthesis network paradigm maybe be extended to include a larger set of more complex facial movements. Three constructed synthesis networks, in particular, stood out as significant contributions beyond the work of Beymer, Shashua, Poggio. These networks were:

- a two-dimensional, 9-example network that synthesizes vertical and horizontal pose movements of the head, shown in figure 2-13. Beymer, Shashua, Poggio did, in fact, construct a two-dimensional network that involved horizontal and vertical pose movement, but the movements represented were only in one direction only (ie from frontal to left and from frontal to up, as opposed to: left to right, and down to up).

- a two-dimensional, 5-example network that synthesizes various mouth expressions involving signicant amounts of occlusion, shown in figure 2-12. Beymer, Shashua, Poggio constructed one- and two-dimensional synthesis networks that involved smile and frown expressions, but these expressions do not contain as much occlusion as expressions involving opening or closing the mouth. The constructed network described in this work involved one parameter that controlled degree of smile, while the second controlled degree of open mouth.

- a 14-example, 5-dimensional network that synthesizes pose movements, eye movements, and mouth movements combined, diagramed in figure 2-19. The largest network Beymer, Shashua, Poggio constructed was an 8-example, 3-dimensional network where one axis controlled horizontal pose, another controlled vertical pose, and the third controlled degree of smile.

Chapter 2 of this thesis describes how each of the above networks was constructed, in addition to other types of networks that were constructed along the way. The description approach is incremental rather than individual, in that network construction is described in increasing layers of complexity, as opposed to describing each of the above networks from start to finish individually.

On the analysis side, the goal was to use the three constructed networks above in an analysis-by-synthesis scheme involving iterative, local, and independent parameter

perturbation, to analyze novel image sequences. For each network used, the analysis algorithm extracted not only the intrinsically modelled parameters, but also a set of six affine parameters as well. To test the robustness of the analysis algorithm, the novel image sequences included moderate differences in facial location, scale, and angle, in addition to lighting and hairstyle differences. Chapter 3 describes the analysis algorithm in more detail, and presents the analysis results obtained. For comparison to the original image sequences, the synthesized image sequences, based on the extracted analysis parameters, are also presented.

Chapter 2

Synthesis

2.1 Overview

In this chapter, we describe the construction of the synthesis modules that are to be used in synthesis of facial image sequences from a set of high-level parameters. We adopt the synthesis network paradigm developed in Beymer, Shashua, and Poggio [5], for several reasons:

- The network paradigm is *example-based*, and hence bypasses the need for any three-dimensional models. Example imagery is also ideal for the purposes of realism.
- The network paradigm is *trainable*, meaning that we can map the examples used for the model to a high-level set of parameters. Both the examples and the parameters may be chosen flexibly by the model designer.
- The network paradigm *learns*, in some sense, to be be able to generalize from the set of images given, to generate novel, intermediate images.

A general modular overview of a synthesis module based on this paradigm is shown in figure 2-1. The collection of example imagery used to model particular facial motions is termed the *example set*. In this case, the example set designer chose to model the mouth opening, so he chose two examples: one with the mouth closed, and the other with the mouth open. After assigning one image to 0.0 and the other to 1.0, the synthesis module tries to learn the mapping from the imposed parameter space to the example set so as to be able to generate a new image that lies, say, at the position 0.512.

In the following subsections, we describe in detail how the example set is chosen and trained, and, subsequently, how novel images are generated. We will also describe specific experiments performed with different types of networks, bearing in mind that our overall goal is to describe the creation of three networks in specific:

- a network that synthesizes vertical and horizontal pose movements of the head
- a network that synthesizes various mouth expressions, and
- a network that synthesizes pose movements, eye movements, and mouth movements combined.

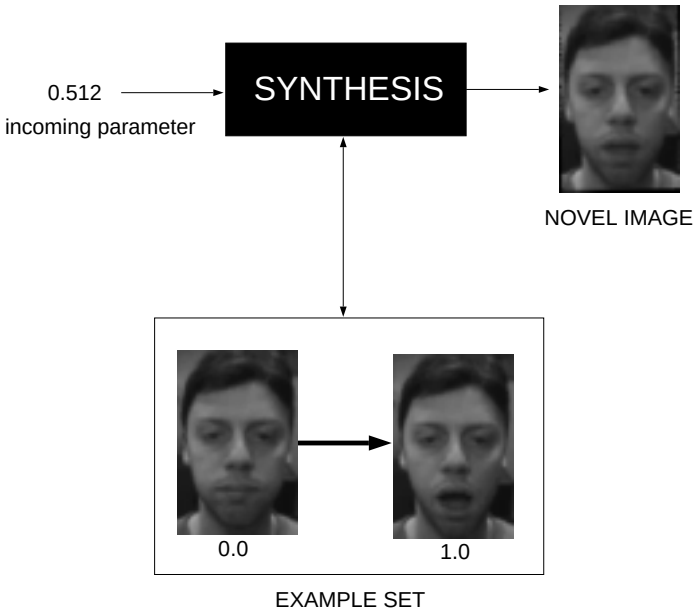


Figure 2-1: General overview of the synthesis module.

2.2 Choosing and Training the Example Set

2.2.1 Example Selection and Parameterization

The first step in the creation of the example set is the selection of the example images and the association of each example with a point in a hand-crafted parameter space. Typically, this entails that the example set designer first record a movie sequence or still shots of his/her face in the particular facial orientations that the example set is to span. Recording a movie sequence is usually preferred over recording still shots because it allows the example set designer to have a larger selection of example images from which to choose. Also, recording a movie sequence and being able to see the movie as it is being recorded allows the example set designer to control his/her facial movements better, which leads to better example set images in the long run.

The next step would be to hand-craft a parameter space, and associate each chosen example image with a point in that parameter space. For example, figure 2-2 depicts five examples arranged in a two-dimensional parameter space where each axis is limited to values between 0.0 and 1.0. One axis denotes degree of *open mouth*, while the other denotes degree of *smile*. Four examples are placed at the corners of the spanned space, while a fifth image lies at the point (1.0, 0.5).

It is important to point out certain features of the parameter space that allow it to flexibly accomodate many synthesis configurations:

Multidimensionality The parameter space is multi-dimensional, in anticipation of complex facial example sets where the designer would want to control many aspects of a face.

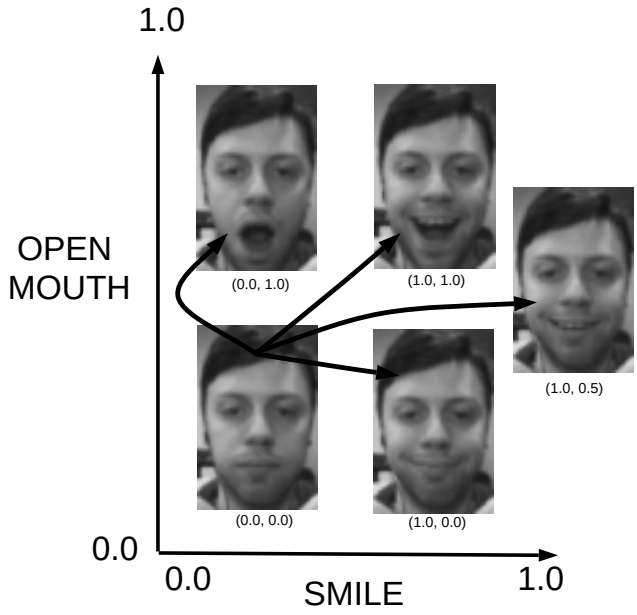


Figure 2-2: A 5-example, 2-dimensional example set in a smile/open-mouth configuration.

Continuity The parameter space is continuous, and not discrete. In figure 2-2, for example, the module generates appropriate examples lying in the whole, continuous space spanned from 0.0 to 1.0 in one dimension, and 0.0 to 1.0 in the other dimension.

Sparseness/Density The example points can be as sparse or as dense as required, and in general there are no rules on how the examples are to be associated with parameter space. In figure 2-2, the designer could have chosen four examples and omitted the fifth example, or he could have placed a few more images in the example set instead.

2.2.2 Correspondence Between Examples

Once the examples and the parameter space are chosen, the next step is to place the examples in correspondence. *Correspondence is perhaps the most critical step in this entire process, as it affords the synthesis module the ability to generate novel, intermediate examples that lie in the space of examples picked by the example set designer.*

Essentially, there are many ways to define correspondence between two images. In this thesis, a *dense*, *pixel-wise* correspondence between two images is defined: for every pixel in image A, we associate a flow vector that refers to the corresponding pixel in image B. The flow vectors are relative to the current pixel position, so for a pixel in image A at position (i, j), the corresponding pixel in image B lies at position

$(i + \Delta x(i,j), j + \Delta y(i,j))$, where Δx and Δy are arrays that contain the x and y components of the flows, respectively. Δx and Δy are the same size as the images A and B.

There are also many ways to obtain such a dense, pixel-wise correspondence between the example images. We adopt the approach used by Beymer, Shashua, Poggio [5], who utilized optical flow algorithms borrowed from the computer vision literature. Specifically, they used the optical flow algorithms developed by Bergen and Hingorani [4], which have the following very important features that should be noted:

- The algorithms are based on Horn and Schunck's optical flow constraint equation ([13]), which assumes that the pixel motion function varies smoothly and continuously. This yields important additional constraints which allow the flow vectors $(\Delta x, \Delta y)$ to be obtained from the *spatial gradients* of both images, and the *time gradient* between them. Due to the approximation of the pixel motion function using a Taylor series expansion, the flow vectors obtained using these techniques are only a *linear* approximation to the actual flow vectors.

- In order to overcome problems in cases where the flow vectors are large, the algorithms are *hierarchical* and *coarse-to-fine*, employing a pyramid-based approach [6] to obtain large correspondences at coarse resolutions, and small correspondences at fine resolutions. Such an approach can, however, introduce *picket-fence* problems.

- The pyramids employed in the algorithms are *laplacian* pyramids, thus affording one to obtain correspondences between images that vary in lighting conditions. On the other hand, laplacian pyramids are known to increase the noise in the image, which may affect the quality of the correspondence vectors obtained.

- No apriori visibility model [28] to handle occlusions is built into the algorithms, so the vectors yielded by the algorithm for a group of pixels in image A that disappear in image B are not guaranteed to be correct. Experimentally, we have found that in such cases the flow vectors returned are usually very close to 0 in magnitude.

- The mapping from image A to image B provided by the flow vectors $(\Delta x, \Delta y)$ is *many-to-one*, in the sense that many pixels in image A may point to the same pixel in image B, causing *overlaps*. The mapping also *contains holes*, in the sense that there are many pixels in image B for which no arrows point to.

From the standpoint of example set design, in which more than two images are involved, a *reference image* is designated, and correspondence between it and the rest of the images in the example set is found. Figure 2-2 depicts the correspondences as symbolic arrows between the reference image chosen (that of a neutral face) and the rest of the images.

2.2.3 Learning the Mapping from Parameters to Examples

The third step is to learn the mapping from parameters to examples.The framework chosen by Beymer, Shashua, and Poggio [5] to accomplish this mapping is regularization theory, described in Girosi, Jones, and Poggio [11]. Regularization theory essentially frames the learning problem as a problem of approximating an unknown function $y = h(x)$ that maps between the example space, x, and the parameter space, y.

It is important to note that Poggio and Brunelli [22] made the extremely crucial observation that, instead of trying to approximate a function $y = h(x)$ that maps between an example space *of images*, x, and the parameter space, y, it is better to try to approximate a function $y = h(x)$ that maps between an example space *of correspondence vectors*, x, and the parameter space, y. The main reason why direct learning of a map from pixels to high-level parameters would not work stems from the discontinuous nature of the underlying map. Learning a map that is not sufficiently *smooth* is hopeless because it requires a very large number of examples. If the examples are too few, any learning algorithm within the framework of regularization theory will not be able to generalize from the available examples.

The particular approximation network that Beymer, Shashua, Poggio use is a *radial basis function with gaussian centers*. Given $(y_i, x_i)_{i=1}^N$, samples of $h(x)$ where the y's are parameters and the x's are correspondence vectors, Beymer et al. construct the following function $f(x)$ that approximates $h(x)$:

$$f(x) = \sum_{\alpha=1}^{n} c_\alpha G(\|x - t_\alpha\|) \tag{2.1}$$

where the G()'s are gaussians

$$G(x) = e^{-\frac{x^2}{\sigma^2}} \tag{2.2}$$

and the $t'_\alpha s$ are arbitrary correspondence vectors termed *centers*.

Essentially, the idea behind a radial basis function architecture is to compare a new correspondence vector x with each of the centers t_α, in this case using a Euclidean distance metric. The outputs of all the comparisons are then weighted in a gaussian fashion, and combined to produce the final approximation to where the original new input vector x lies in the space of vectors t_α.

The learning stage of a radial basis function architecture consists of the specification of three sets of variables:

Choosing the location of the centers t_α

Beymer, Shashua, and Poggio adopted a learning architecture where one example correspondence vector x_i is associated with each center t_α, so the approximating function $f(x)$ becomes :

$$f(x) = \sum_{i=1}^{N} c_i G(\|x - x_i\|) \tag{2.3}$$

```
for (i = 0; i < N; i++) {
     acc = 0.0;
     for (j = 0; j < N; j++) {
       if (i != j) {
         norm = ‖x_i − x_j‖;
         acc = acc + norm;
       }
     }
     σ_i = k(acc/(N−1));
}
```

Figure 2-3: **SIGMA DETERMINATION** algorithm.

N denotes the total number of example correspondence vectors used to train the network.The number of example correspondence vectors, of course, is equivalent to the number of example images used. If there are four example images, there are four correspondence vectors: three vectors denoting correspondence between the chosen base image and the other images, and one reference, zero correspondence vector denoting correspondence between the base image and itself. Each of these correspondence vectors is associated with a x_i in the formula above.

Choosing the σ's of the gaussians

The sigmas of the gaussians, which denote the width of their influence, are determined using an *average inter-example distance strategy*. For each gaussian, the average distance between its associated correspondence vector and the other correspondence vectors is found. The final sigma value associated is chosen to be some fixed constant k times the resulting average. Figure 2-3 depicts the pseudo-code to determine the sigmas for the gaussians. k is usually set to 0.5.

Determining the coefficients c_i

The c_i coefficients are chosen in a manner that minimizes the empirical error between the approximating function $f(x)$ and the sample points $(y_i, x_i)_{i=1}^N$ that are chosen by the example set designer.

If we substitute all the sample pairs into the equation

$$f(x) = \sum_{i=1}^{N} c_i G(\|x - x_i\|) \tag{2.4}$$

we obtain the equation

$$Y = CG \tag{2.5}$$

where

$$Y = [\ y_1 \quad y_2 \quad \dots \quad y_N\ ] \tag{2.6}$$

$$C = [\ c_1 \quad c_2 \quad \dots \quad c_N\ ] \tag{2.7}$$

and

$$G = \begin{bmatrix} G(\|x_1 - x_1\|) & G(\|x_2 - x_1\|) & \dots & G(\|x_N - x_1\|) \\ G(\|x_1 - x_2\|) & G(\|x_2 - x_2\|) & \dots & G(\|x_N - x_2\|) \\ \vdots & & \ddots & \vdots \\ G(\|x_1 - x_N\|) & G(\|x_2 - x_N\|) & \dots & G(\|x_N - x_N\|) \end{bmatrix} \tag{2.8}$$

The coefficients C are then determined by computing

$$C = YG^+ \tag{2.9}$$

where G^+ is the pseudoinverse of G.

The Dual Representation for Synthesis

In the case of the synthesis module, however, it is helpful to rewrite the approximation function into its *dual representation*. Continuing from equation 2.3, we have

$$y(x) = Cg(x) \tag{2.10}$$

where

$$g(x) = [\ G(\|x - x_1\|) \quad G(\|x - x_2\|) \quad \dots \quad G(\|x - x_N\|)\ ] \tag{2.11}$$

Substituting

$$C = YG^+ \tag{2.12}$$

into 2.10, we obtain

$$y(x) = YG^+g(x) \tag{2.13}$$

Gathering the terms not related to Y together, we have

$$y(x) = \sum_{l=1}^{N} b_l(x)y_l \tag{2.14}$$

where

$$b_l(x) = (G^+)_l g(x) \tag{2.15}$$

Equation 2.14, which represents the dual representation of equation 2.3, is arguably the most central equation for synthesis. If the space of correspondence vectors is associated with y and the space of parameters with x, then equation 2.14 represents any vector as a linear combination of the N example correspondence vectors y_l. The

coefficients of the combination, $b_l(x)$, depend nonlinearly on the parameter x whose correspondence is desired.

2.3 Synthesizing Novel Images

After choosing the images, designing the parameter space, establishing correspondence, and learning the map from parameters to correspondence, the last step is to synthesize new intermediate images for any particular set of inputs that lie within the parameter space chosen.

Because the decision was made initially to learn the map from parameters to correspondence vectors rather than images themselves, the process of creating new, intermediate images needs to be solved in two stages. The first stage is the stage when the trained network synthesizes the appropriate correspondence, and the second stage is a *rendering* stage, when the synthesized flow and the example images are used to render the new image.

2.3.1 Synthesizing Flow

Synthesizing a new flow follows from equation 2.14. For any new parameter x, the network will:

- compute G^+, where G is defined as in 2.8
- compute $g(x)$, which is defined in 2.11
- compute the nonlinear kernels $b_l(x) = (G^+)_l g(x)$
- combine the kernels linearly with the example correspondence vectors y_l to produce the new, intermediate correspondence vector y according to equation 2.14

2.3.2 Rendering Images at the Synthesized Flow

To render the synthesized correspondence vector, a morphing technique [3] [8], involving a combination of pixel *warping* and pixel *blending* is used. Figure 2-4 depicts a synthesis pipeline in the case of a one-dimensional example set with two examples. After the new correspondence vector, which is depicted as the solid black arrow vector, is synthesized in step 1, a flow re-orientation stage re-orients the synthesized flow so that that it originates from each of the example images in the network. This allows all the example images in the network to be warped along the re-oriented flows to the position specified by the synthesized flow. Finally, a blending stage blends the warped images to produce the final synthesized image. In the sections that follow, we describe each of the stages in detail.

Flow Re-orientation

In the case of the reference example in the example set, no re-orientation is needed, since the synthesized flow already originates from the reference image in the example

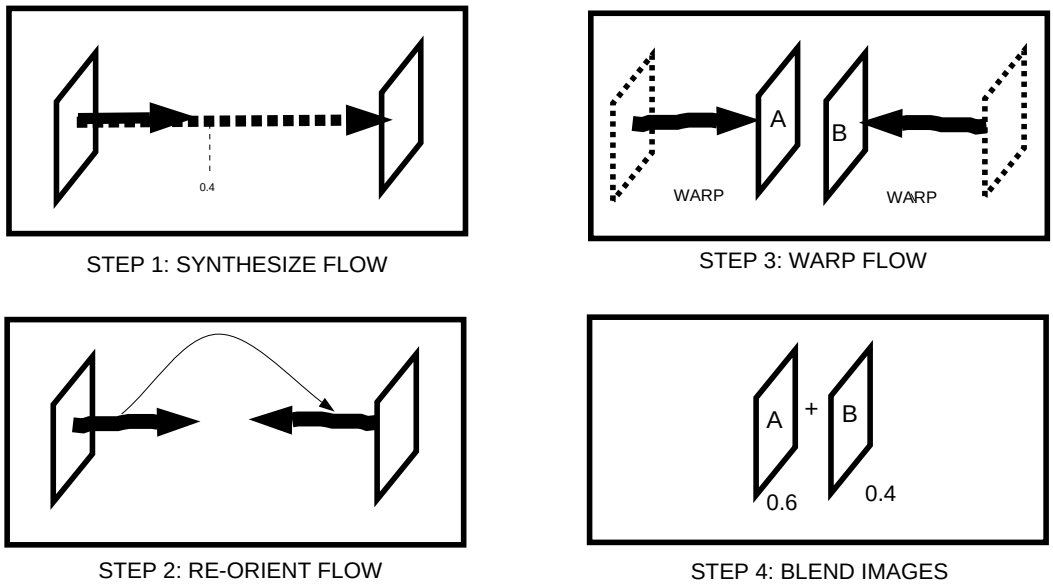


Figure 2-4: Synthesis pipeline for the case of two examples set in one dimension

set. This is due to the fact that all the example correspondences are defined initially between the reference example and the rest of the examples. In the case of the other examples, however, re-orientation is needed before the example images may be warped.

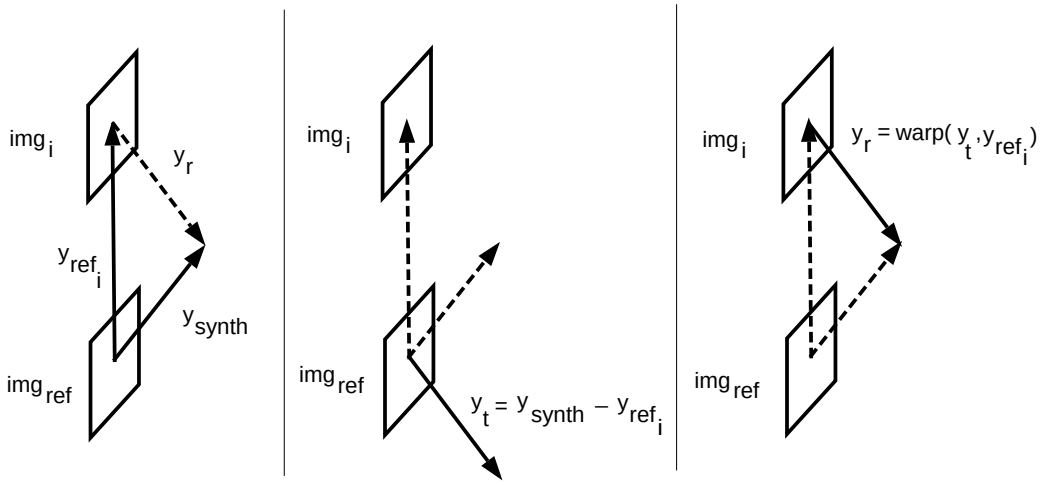
One possible method to re-orient flow vectors is shown in figure 2-5. The goal is to re-orient the synthesized flow y_{synth} so that it originates from img_i instead of img_{ref}. The synthesized flow y_{synth} is first subtracted from y_{ref_i} to yield y_t. y_t will contain the correct flow geometry, but will originate from the reference example img_{ref} rather than the desired example image img_i. To move y_t into the correct reference frame, the flow vector is warped along the original reference flow y_{ref_i}. The resulting flow y_r is the desired re-oriented flow.

Re-orientation is performed for all examples in the example set, with the exception of the reference example, since the synthesized flow already originates from it. In the case of a one-dimensional network with two examples, this process is done once, while in the case of a two-dimensional example set with four examples, this process is done three times.

Warping the Examples

The example images are all subsequently warped along the resulting re-oriented flows. In this case, a *forward warp* algorithm, shown in figure 2-6, is employed to move the pixels along the flow vectors. The pixel destinations are rounded to the nearest integer, since the addition of the flow vectors may lead to non-integer destinations.

The forward warp algorithm used does not explicitly treat *pixel overlaps* in any special way, since there is no apriori visibility or occlusion model built into the algo-



$\overline{y}_t = \overline{y}_{synth} - \overline{y}_{ref_i}$
$\overline{y}_r \leftarrow$ **FORWARD_WARP_FLOW_FIELDS** $(\overline{y}_t,\ \overline{y}_{ref_i})$

Figure 2-5: **RE-ORIENTATION** algorithm.

```
for (j=0; j < h; j++)
     for (i=0; i < w; i++) {
        x = i + Dx[j][i];
        y = j + Dy[j][i];
        x1 = ROUND_TO_NEAREST_INT (x);
        y1 = ROUND_TO_NEAREST_INT (y);
        if (x1,y1) are within the image
           Out[y1][x1] = In[j][i];
     }
```

Figure 2-6: **FORWARD WARP** algorithm

rithm, unlike [8] and [19]. The order of the warp, as shown in figure 2-6, is a *top-down, left-to-right* order, which is an order that does not take into account particular visibility constraints.

Particular types of *holes* in the correspondences are, and must be, explicitly treated, however, since they lead to noticeable degradations in the quality of the final images that are synthesized. In particular, holes due to local image expansion and inaccuracies in the optical flow algorithms due to lack of discriminating texture usually lead to small *specks* in the warped image. These holes are identified and eliminated as in [8], by filling the warped image with a reserved "background" color. For those pixels which retain the background color after the warp, new colors are computed by interpolating the colors of the adjacent non-background colors.

Blending

The final stage of the rendering process is the *blending* stage, when all of the warped images are blended together to produce the final image. Blending refers to multiplying each image with a blending coefficient, and then adding all the scaled images together to form the final image. The blending coefficients chosen are exactly the same as the coefficients $b_l(x)$ (see formula 2.14) that are computed by the network in its synthesis of the new correspondence vector. It is necessary, however, to normalize all the coefficients beforehand so that their sum is exactly 1.0. This is done to reduce the chances that the addition of all the scaled images produces pixel values greater than 255 (in the case of 8-bit grey-scale).

Re-Scaling

Sometimes, after combining all the intermediate images using blending, it is also necessary to re-scale the final image's pixel values so that its average pixel value matches the average pixel value of the example images. This corrects for any shifts in the average pixel value of the final synthesized image, which leads to noticeable flickering in sequences since the average pixel value affects the brightness of the image.

Discussion

It is vital, given the importance of rendering to this whole process, to flesh out certain aspects of the rendering algorithm.

Firstly, the outputs of such a synthesis scheme are always *matched* to the actual example images for input parameters that lie *at the positions of the example images themselves*. This is due to the particular choice of training coefficients, which are chosen so that, for input parameters that are equivalent to the parameters of the example correspondences, the synthesized correspondences are matched to the example correspondences. And, since the same coefficients are used to blend the example images, the synthesized images are also matched to the example images.

Secondly, the combination of warping and blending is more powerful than either technique on its own. Blending alone can generate intermediate images, but a sense of *movement* between images will be lacking. Warping alone will expose the deficiencies

of the optical flow algorithms, particularly its linearization errors: the flow estimates at each pixel are, for the particular class of algorithms developed by Bergen and Hingorani [4], only a *linear* approximation to the actual flow. As a result, warping from one image by itself will lead to suitable intermediate images *only* when the parameters are close to the parameters of the example image, but will lead to incorrect images as the parameters move farther away. Warping from all the examples combined with weighted blending, however, eases the linearization errors because, as the parameters move farther away from one example image, the pixel motion and pixel values of another example image begin to take effect. In this sense, a linear flow only needs to span half the distance between examples, because another linear flow from another example (in our case, a re-oriented version of the same flow) will play the dominant synthesis role for parameters close to that example.

If the linear flows do not sufficiently span the example space, such as when the optical flow algorithms fail to establish reasonable correspondence, noticeable *shadows regions* start to appear: regions of the face that are blended into other regions without realistic motion between them. Consequently, care must be taken to pick example images that will result in flow that is good enough to create realistic synthesis. An important and commonly used technique to improve correspondence between images will be described later.

Overall, however, it is extremely heartening that a technique that combines warping and blending can lead to results that are not only good for cases in which the self-occlusions of the face are *present*, as will be shown shortly, but also in cases in which the self-occlusions are *large* (as when the mouth is opening and closing.) Furthermore, it is even more heartening that extending this technique from the simple 1-dimensional cases to n-dimensional cases with a considerably larger set of examples also works; that is, warping and blending a larger set of images also leads to good results, and does not lead to an increase in noise.

In the next section, we will describe synthesis experiments involving different types of networks.

2.4 1-Dimensional Networks

2.4.1 Experiments

A number of synthesis experiments were performed with a 1-dimensional example set of two examples lying at 0.0 and 1.0, respectively. These sets of experiments were designed to test the feasibility of the proposed synthesis technique on a wide variety of common facial movements, such as pose movements, mouth movements, and eye movements. Figure 2-7 shows the results of two such experiments. In the top case, the parameter denotes degree of smile, while in the lower case, the parameter denotes a degree of pose. In both cases, the leftmost and rightmost images are original examples, while the images in between are synthesized.

By and large, it was found that a large number of the facial movements that involved the mouth and eyes, and hence, not a lot of occlusion, could be handled using

Figure 2-7: 1-dimensional synthesis network experiments.

this synthesis technique, leading to synthesized intermediate images that looked very realistic. Images in which there was small pose movements, as well, could also be handled well using this technique. In cases where the occlusions were large, such as when pose movement was large, the intermediate images did not look very realistic, leading to the *shadow* phenomenon described earlier.

2.4.2 Experiments with Composed Flow

To alleviate the problems cause by self-occlusions and large displacements, experiments were performed in which optical flow was computed incrementally through the use of intermediate images that lie in between the two examples designated for inclusion in the example set. In other words, instead of taking still shots initially, a sequence is taken, and the intermediate images used to obtain better optical flow.

The way this is done is through the computation of optical flow between each and every pair in the sequence, and the gradual accumulation of these flows in one, final composed flow vector, as depicted in the pseudo-code in figure 2-8. To illustrate, consider two frames A and C, for which a direct optical flow computation will lead to bad correspondence, maybe due to the fact that the images are just too far apart. Suppose that there was an intermediate image B. To obtain *composed* flow from A to C, we first compute flow from A to B, and from B to C, directly. The flow from B to C is then *warped backwards* along the flow from A to B, so as to put it in the same reference frame as that of the flow from A to B. Now that both flows are in the same reference frame, they can be added to yield the composed flow from A to C. If C itself was not the final image in a sequence, but rather an intermediate image, then this whole process is repeated again with respect to D, the next image.

Backwards warping is similar to forward warping, except that flow vectors are

```
y_c ← FINDFLOW_FWD (seq[0], seq[1] );
for i=1 to num_images-1 do {
      y_t ← FINDFLOW_FWD ( seq[i], seq[i+1]) ;
      y_w ← BACKWARD_WARP_FLOW_FIELDS (y_t, y_c);
      y_c = y_c + y_w;
      }
```

Figure 2-8: **COMPOSED FLOW** algorithm

traversed to reach a source image or source flow vector which is warped backwards. Pseudo-code for the backwards warp used in this thesis is shown in figure 2-10. It is important to note that, in cases when the flow points to a non-integral region of the source image or flow, a bilinearly weighted combination of the four closest pixels is used.

Experiments performed using composed flow have shown that this technique is capable of eliminating many of the occlusion problems and distance problems encountered in earlier attempts to establish correspondence. The synthesized column of faces on the left in figure 2-9 is obtained using composed flow, as compared with the one on the right which is obtained using direct flow. As can be seen by comparing the third frames from both sequences, composed flow reduces shadow effects significantly.

The use of composed flow, while serving to alleviate correspondence problems, *does constitute, however, a significant shift in our modelling paradigm*, since the model to be used for synthesis is now composed not strictly of images, but rather of sequences. Technically, however, the intermediate images in the sequences are not used beyond the correspondence computation stage, so our model of a face is still composed of images, the flow between them, and a learning network to map from high-level parameters to flow.

The use of sequences instead of still-shot examples may also be viewed negatively considering that we may be violating one assumption underlying the motivation of this work: namely, that the *available memory is limited*, and that model-based coding is resorted to for its higher rates of compression. However, the construction of a model for a human face needs to be done only once, and does not need to done on the same memory-limited computer that is going to be used for analysis. Furthermore, after the intermediate images in the sequences are used to compose flow, they may be discarded.

It is important to note that the use of composed flow causes problems of its own, and that there is an important trade-off to be made: sometimes, especially when the original direct flow is good to begin with, composed flow can actually increase the noise in the flow vectors, leading to rendering that is actually worse. The errors in the flow vectors usually accumulate due to the repeated warping that is done when flow composition is performed. When there are lots of frames in between two examples which have not moved very far apart, it is preferable to use direct flow.

In the following sections, a mixture of composed and direct flow computations are made to obtain the correspondences between the images in the example set, with

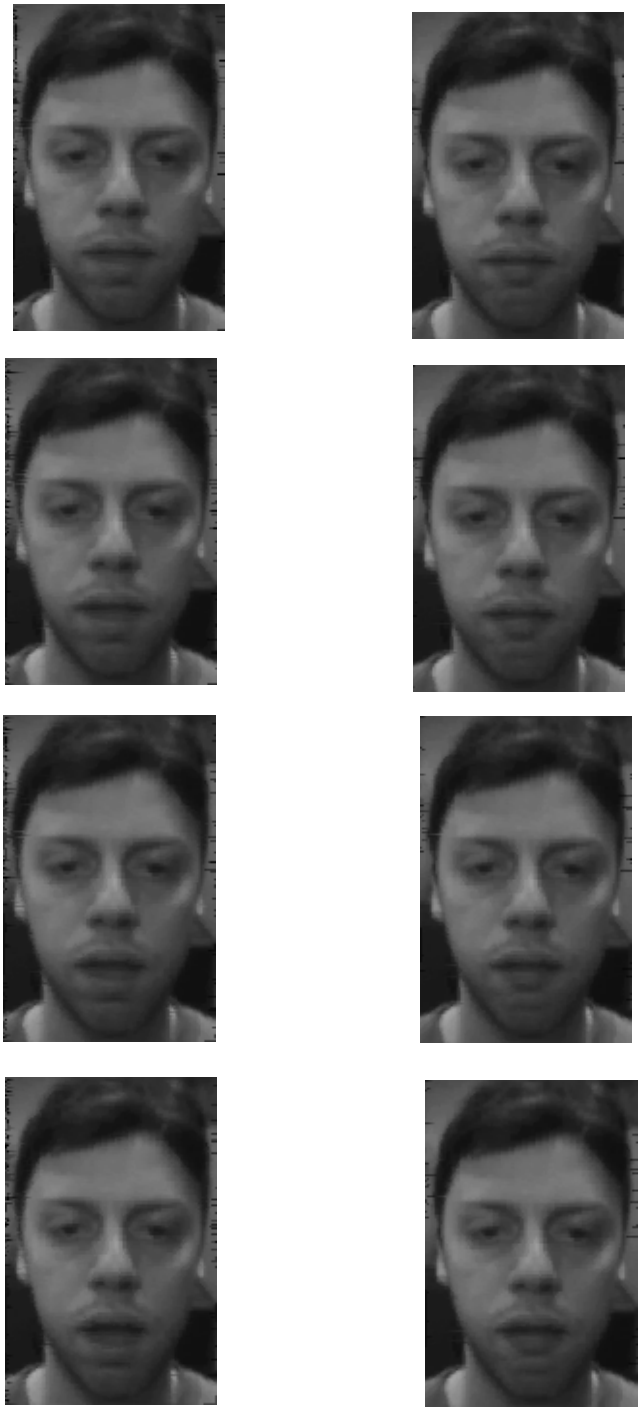


Figure 2-9: 1-dimensional, 2-example synthesis of an opening mouth. The left sequence is obtained using composed flow, while the one on the right is obtained using direct flow.

```
for (j=0; j < h; j++)
     for (i=0; i < w; i++) {
        x = i + Dx[j][i];
        y = j + Dy[j][i];
        Out[j][i] = BILINEAR (In, y, x);
     }
```

Figure 2-10: **BACKWARD WARP** algorithm

composed flow predominating in most circumstances.

2.5 2-Dimensional Networks

Synthesis experiments with 2-dimensional networks were also performed in an attempt to synthesize various face motions, such as pose movements, eye movements, and mouth movements. Technically, the 2-dimensional learning, synthesis, and rendering algorithms are exactly the same as their 1-dimensional counterparts. *2-dimensional networks are significantly more important than 1-dimensional networks, however, because they are extremely well-suited to model both pose movements and eye movements.* In both cases, the 2-dimensional synthesis parameters control movement along the vertical and the horizontal axes. In figure 2-11, synthesis examples from a 4-example synthesis network for pose are shown, where one axis is upward head movement, and another axis is leftwards head movement.

Several experiments were also performed with 2-dimensional networks involving mouth movements, and one such important result is displayed in figure 2-12. In this case, one axis denotes the degree of the mouth's openness, while the other axis denotes the degree of the mouth's smile. The synthesis examples are obtained from the same 2-dimensional, 5-example example-set that was shown in figure 2-2. Although the 2-dimensional network is not perfectly suitable for capturing all the nuances of facial mouth movement, neverthess, as the figure illustrates, it serves to depict the strength of our technqiue in spanning a large space with non-trivial, novel, intermediate examples that involve a large amount of facial self-occlusion. This is particularly important given the fact that no apriori visibility or occlusion model was built into the optical flow and synthesis routines.

The 2-dimensional, 5-example network shown in figure 2-12 constitutes one of the networks that will be used in Chapter 3 to analyze novel image streams.

2.6 Composition of Local Networks

If a larger 2-dimensional space needs to spanned, more examples can be added to the network. For example, if the pose network shown in figure 2-11 is to be expanded to include examples with the head looking downwards and with the head looking

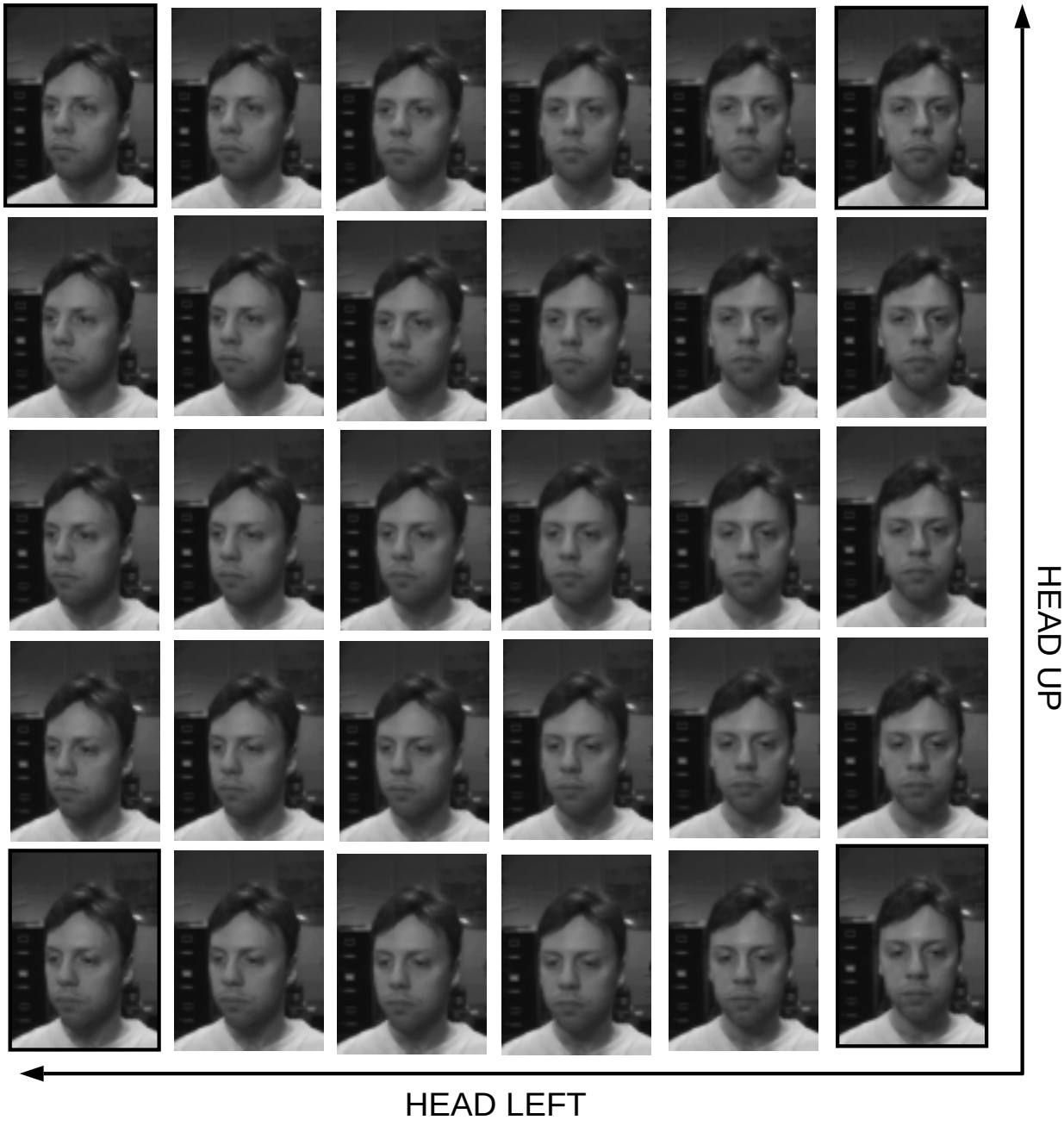


Figure 2-11: 2-dimensional, 4-example synthesis network for pose. The examples at the four corners are original images.

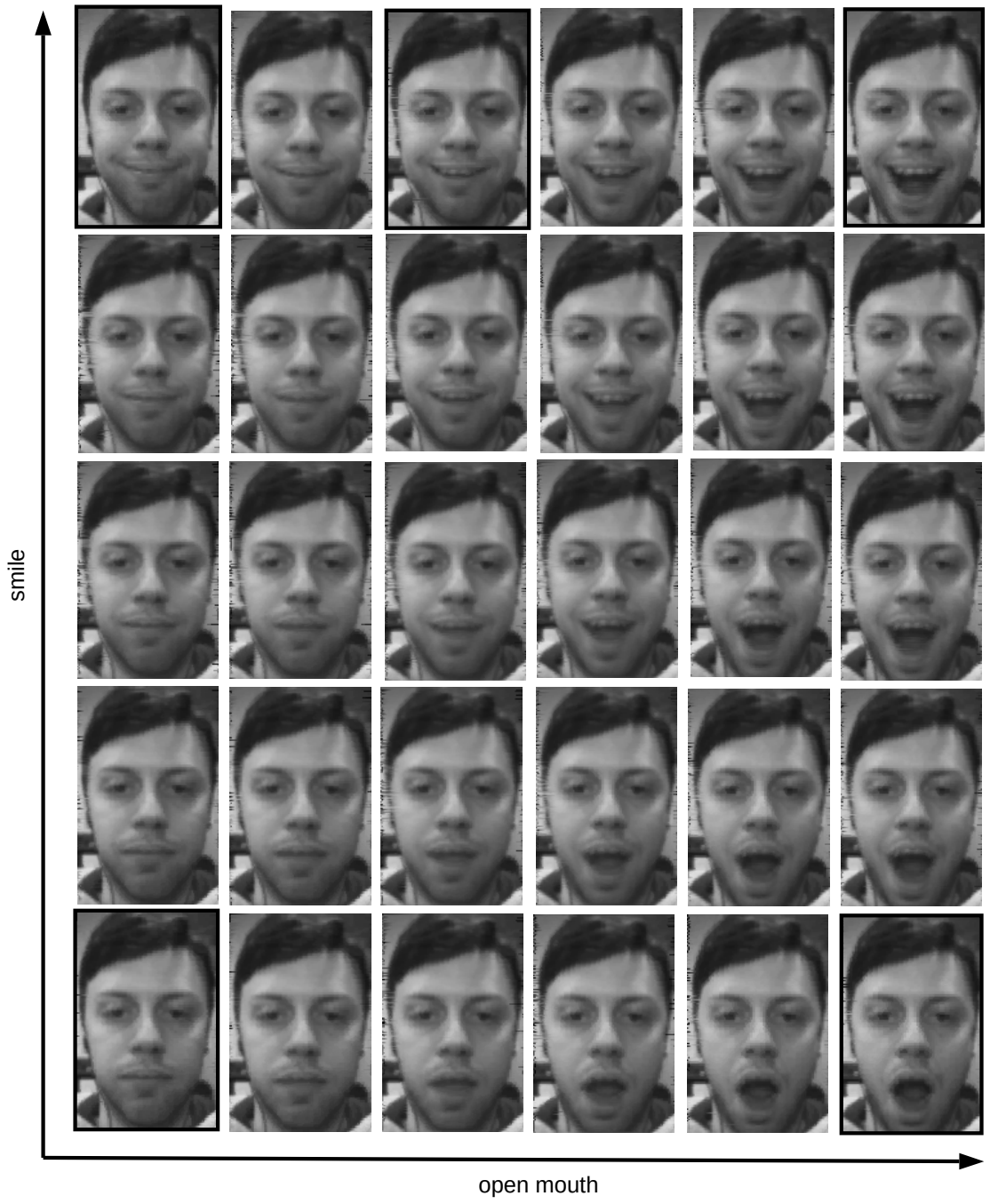


Figure 2-12: 2-dimensional, 5-example synthesis network for facial expressions with occlusion. The original images are high-lighted with darker borders.

towards the right, it is possible to add 5 more examples, creating a 3-by-3 network, as shown in figure 2-13.

Such a 3-by-3 network, however, may be viewed as four 2-by-2 networks that share a common set of example images along the adjacent edges. Instead of traversing one large network space, smaller, *local* network spaces are traversed instead, and a navigational mechanism is utilized to determine which local network is currently activated. Experiments were performed with exactly such a set of 4 composed local networks denoting horizontal and vertical pose movement, and some of the synthesized results are shown in figure 2-14. The navigational mechanism used in this case performs a horizontal and vertical threshold check based on the input parameters to check which network is activated.

There are several major advantages of using such a network composition technique. Firstly, *composition is natural*, at least within a synthesis framework based on morphing. If the 2-dimensional space is large, chances are that the intermediate examples should only be determined from the example images that are the closest. For example, synthesizing intermediate pose examples in which the head faces left should really only rely on the example images of the face looking left and those look straight ahead. The example images of the head looking to the right should not factor into the synthesized intermediates at all. If all the examples belong to the same network, then they all factor into the creation of any intermediate example.

Secondly, *composition maintains constant computation complexity*. No matter how large an example set space becomes, if one synthesizes only from the four closest examples, then the computational complexity remains constant. The only price to be paid is the price of having to decide which network to activate, which is not as computationally intensive as having to warp and blend from a large number of examples.

Thirdly, *composition improves final image quality*. By using only the most relevant images for synthesis, the final image quality is improved. Image quality tends to decrease when a large number of examples are warped and blended together, due to the accumulated errors.

The key to composition, of course, is the fact that adjacent networks share the same examples along the adjacent edges. While this allows the synthesized *images* along the transitions between the networks to be seamless, it does present the problem of a discontinuity in the *motion velocity or direction*: the paths and the speeds of a face, for example, may change as one moves from one local network to another. Rectifying this problem, however, was beyond the scope of this thesis.

The 2-dimensional, 9-example network consisting of four smaller, local networks shown in figure 2-12 constitutes one of the networks that will be used in Chapter 3 to analyze novel image streams.

2.7 N-dimensional Networks

It is also possible to extend the 1-dimensional and 2-dimensional cases to arbitrary N-dimensional cases, where each example in the network is assigned to a point in an

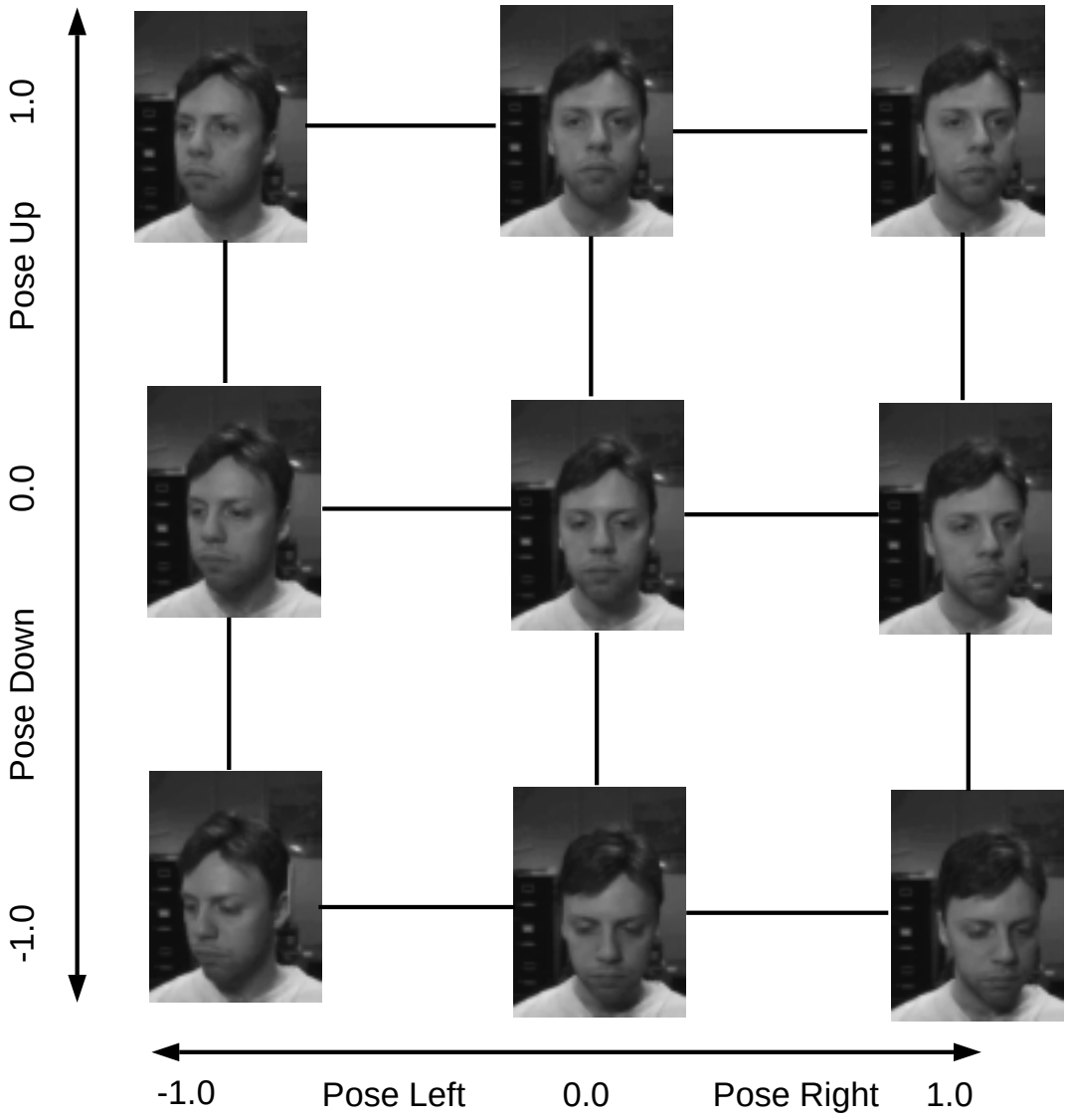


Figure 2-13: The examples for a 3-by-3 network involving pose movements in all directions.

Figure 2-14: Some intermediate examples generated from the synthesis network of the previous figure.

N-dimensional parameter space.

Such an N-dimensional configuration was explored in the context of trying to create a synthesis network that involved eye movements, mouth movements, and pose movements combined. The eyes were modeled using a 2-dimensional network of 6 examples. One mouth dimension was chosen indicating the degree to which the mouth was open, and one pose dimension was chosen to indicate the degree to which the head was rotated from frontal to the right.

The entire configuration is shown in figure 2-15. The top-left 6 images belong to the eye positions with a closed mouth and a frontal pose; the 6 images in the lower-left belong to the eye positions with an open mouth and a frontal pose; those in the top-right belong to the eye positions with a closed mouth and a rightwards pose; and, finally, those in the bottom-right belong to the eye positions with an open mouth and a rightwards pose. There were a total of 24 examples in the entire network.

Synthesis of an N-dimensional network is the same as that of a 1-dimensional or a 2-dimensional network: for an input set of paremeters, the synthesis module uses equation 2.14 to synthesize a new flow lying in the large N-dimensional space. To render, re-oriented flows are generated from all the examples to the synthesized flow, in order to allow the texture from the various examples to be warped and blended accordingly.

Experiments performed with such a network, however, showed mediocre results at best for several reasons:

- Firstly, it became obvious that if more dimensions were to be added, the number of examples needed would grow considerably. Adding a mouth dimension to a set of 6 eye examples, for instance, led to the need for another set of 6 eye examples at the new mouth position. Is it possible to exploit the inherent independence of eye movements from mouth movements to reduce the number of examples that need to be taken? Reducing the number of examples would

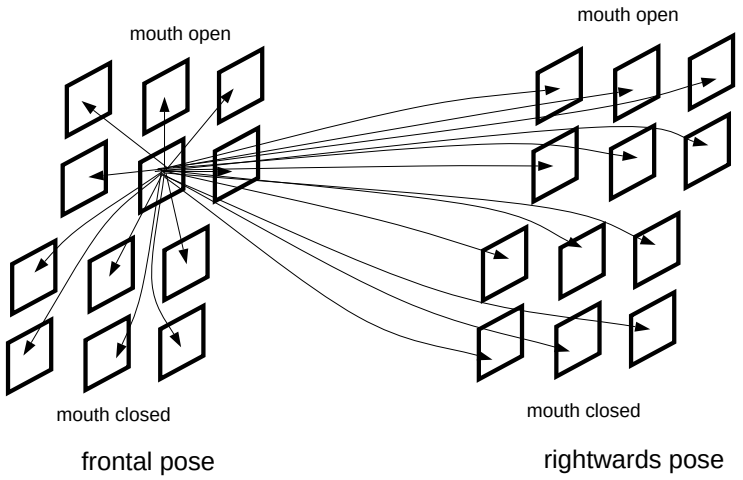


Figure 2-15: A 5-dimensional, 24-example network configuration.

not only reduce the memory requirements but also reduce the difficulty and tediousness involved in capturing these examples to begin with.

- Secondly, in a large N-dimensional network where all the examples are warped and blended to achieve the final result, a lot of time is spent on warping and blending examples that really do not contribute much to the final result, due to the fact that their respective blending coefficients are of small magnitude. Warping and blending such a large number of examples also serves to degrade the final output, making it look less sharp. Is there a way to partition the space effectively to achieve synthesis results that only involved the relevant images, and would thus produce results that looked much better, much as the pose space was partitioned in figure 2-13 using composition of local networks?

- Finally, and most importantly, it was noticed that placing such a large number of examples created a problem of *interference*: for a particular pose and mouth orientation, for example, synthesizing a new eye position did not only move the eyes, but also involved siginificant pose and mouth changes. The network had not learned to completely align the space of correspondence vectors with the imposed parameter space.

2.8 Regional Networks

To alleviate some of the problems associated with the large number of required example images needed whenever a new dimension is added to a synthesis network, experiments were performed involving the creation of separate, *regional* networks for different parts of the face that move independently of each other. To understand how regional decomposition can reduce the need for extra examples, consider the need to

model 6 eye positions and 4 mouth positions. Without regional decomposition one would need 6x4, or 24 examples, whereas with regional decompostion one would need only 6+4, or 10, examples.

Regional decomposition needs to address two issues: how to specify which regions each network controls, and how to combine the synthesized outputs of all the regional networks back together again.

A *mask-based* approach was adopted to specify which regions of the face each network controls. At the outset of the example set selection, the example set designer uses a special tool to "mask out" which region of the face each network controls. The mask produced by the tool is essentially a binarized image. During synthesis, a navigational mechanism first determines which parameters have changed relative to previous parameters, and identifies which regional network is activated. The parameters associated with that regional network are then used to synthesize an image. The mask associated with that regional network is then used to extract the appropriate portion of the synthesized image.

To combine the masked regions back together again, a simple *paste* approach was adopted, where the regions are pasted on top of a *base image* of the face. This approach works extremely well if the motion is contained *within* the regions themselves. Ideally, one would want to *blend* the regions onto the base image.

Regional networks were constructed for left eye motions, right eye motions, and mouth motions, as shown in figure 2-16. The regional left and right eye networks were composed of the same six images placed in a 2-dimensional arrangement, but the masks for each was different, as shown. The mouth regional network consisted only of two examples to model an opening mouth. The mask for the mouth network consisted of all the pixels not contained in the left and right eye regional networks; this approach enables one to avoid creating a mask with a more explicit segmentation of the mouth region, which is hard to do because mouth movements affect a large portion of the face. The masked outputs of each regional network are pasted onto the base image shown in the center of the figure.

The gain in possible eye-mouth configurations given the number of example images is now much higher than in a standard approach not involving regional networks. Using only 6 original eye images (since the left and right eye regional networks use the same images) and 1 additional open-mouth image (since the reference image is the same for all the regional networks), various combinations of eye-mouth positions may be synthesized, as shown in figure 2-17. This added flexibility is also a boon for the example set designer, who needs fewer example images to build the desired synthesis model. On the other hand, the example set designer now needs to specify the mask regions.

2.9 Hierarchical Networks

A new synthesis approach was introduced to alleviate the problems associated with *interference*, which emerged in the attempts to create a network which could synthesize eye, mouth, and pose movements combined. The interference problem arises because

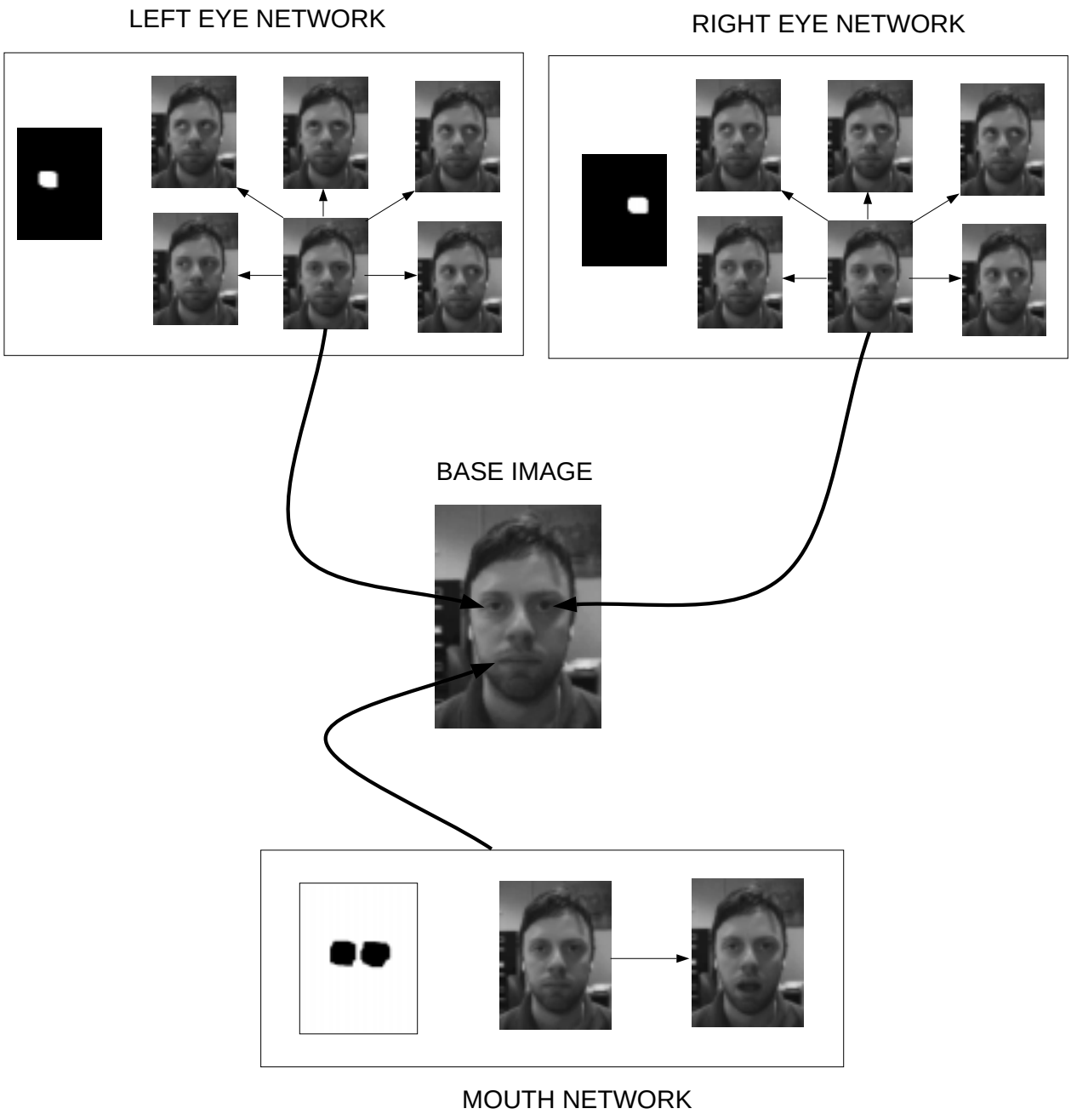


Figure 2-16: Construction of a 7-example, 5-dimensional regional synthesis network controlling mouth movement and eye movement.

Figure 2-17: Synthesized images generated from the network in the previous figure.

equation 2.14 synthesizes new flows by linearly combining the example correspondences already established by the example set designer, *which might not be suitably orthogonalized with respect to certain motions that the example set user might wish to move along.* In the standard N-dimensional configuration, it was noticed, for example, that synthesizing a different eye position for a constant pose would result in pose movement nevertheless. Similarly, synthesizing a new mouth position would cause unwanted pose movement as well.

In a larger sense, interference is detrimental because it violates an *inherent hierarchy* between synthesis networks: eyes and mouth networks should be subnetworks within a larger pose network. Synthesizing new eye positions and mouth positions should not interfere with the global pose, while changing pose will necessarily affect the images synthesized by the eye and mouth subnetworks. Consequently, there was a need for a new synthesis approach which attempted to encode this new notion of hierarchy between networks.

2.9.1 A New Synthesis Approach

Figure 2-18 depicts the various stages of this new synthesis approach for an illustrative example involving a 2-dimensional, 4-example network encoding pose along one dimension, and mouth movement along the other. Figure 2-18 a) depicts the standard synthesis configuration for such a network.

In the new approach, as shown in figure 2-18 b), the left-most and right-most images and flow are viewed as two individual, 1-dimensional *mouth subnetworks* that only control mouth position for a given pose. All mouth subnetworks should have the same number of images, the same number of dimensions, and be completely devoid of any pose movement.

In addition, a new set of flows termed *cross-flows* are obtained, which link the images of the mouth subnetworks together. In figure 2-18 b), the cross-flows are shown as the dotted arrows. In some cases, the cross-flows are already known, as in the case of the bottom cross-flow in the figure, which is the same as the flow established by the example set designer in the standard configuration. For cases when the cross-

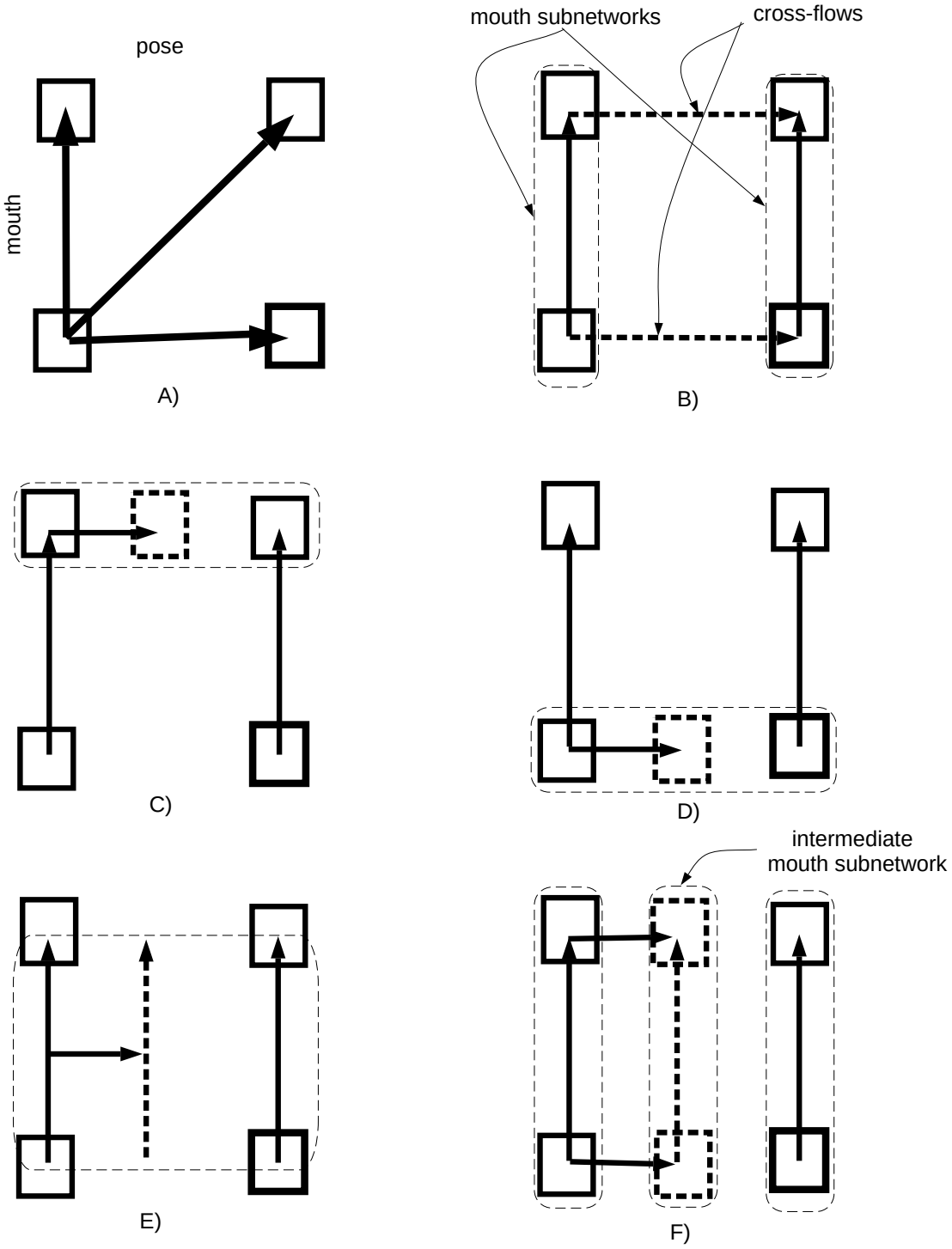


Figure 2-18: The stages of the new hierarchical synthesis approach for a 4-example, 2-dimensional example set.

flows are not known, one method to obtain them is to use the standard synthesis network of figure 2-18 a) to synthesize images *along the cross-flow direction*, and then to *compose the flow* between all those synthesized images to create the final cross-flow. Of course, the synthesized intermediate images will themselves exhibit the interference effects that we are trying to eliminate, but computing composed flow eliminates these effects because it is cumulative: as long as the two endpoint images belong to the respective mouth subnetworks, flow composition will "forget" the intermediate interference effects.

The goal of the new synthesis approach, when the pose is changed, is now to morph the entire mouth subnetworks to create an intermediate mouth subnetwork that lies at the desired pose, as shown in figure 2-18 f). Synthesizing a new eye-mouth subnetwork consists of two steps:

- The first involves synthesizing the new, intermediate *images* that belong in the new, intermediate subnetwork, as shown in figures 2-18 c) and figure 2-18 d). The synthesis of the new images proceeds along the respective cross-flow vectors. Essentially, temporary 1-dimensional synthesis netorks are created, where the corner images are the corresponding images in the mouth subnetworks, and the flow vector is the cross-flow vector. Synthesis of the intermediate images proceeds in the standard manner.

- The second step, shown in 2-18 e), involves the synthesis of the new, intermediate *flow* tying the images within the new, intermediate subnetwork together. In this case, a temporary network is created *in which the endpoints are not images, but the two flows from the corner mouth subnetworks.* These flows are warped and blended to produce the intermediate flow that ties the images within the intermediate subnetwork together.

In this manner, the new synthesis approach synthesizes a new, intermediate subnetwork whenever changes in pose are made. When changes are required to the mouth positions for a particular pose, new images are synthesized *within* the intermediate subnetwork. To maintain continuity in the synthesized image stream, if changes in pose are required once a particular setting of mouth positions are chosen, then the synthesis module not only has to synthesize the new mouth subnetwork at the desired pose, but, within that network, it has to synthesize an image at the correct mouth setting.

2.9.2 Combined Eye, Mouth, and Pose Synthesis

The new synthesis paradigm was applied to the 24-example 5-dimensional network that involved eye, mouth, and pose movements, shown in figure 2-15. Firstly, the 12 examples involving eye and mouth movements for the frontal pose were reduced to a 7-example network composed of two regional networks for the eyes and mouth. A similar 7-example network was also created for the eye and mouth movements at the rightwards pose. The regional networks for both frontal and rightwards pose are

shown in figure 2-19 a). Using regional networks thus reduced the total number of examples needed, from 24 to 14.

The next step in the creation of the eyes-mouth-pose network, shown in 2-19 b), was to compute the cross-flows linking the images and flow between the two regional networks. This was done using the technique described in the previous section, involving the synthesis of intermediate images and composing flow.

The third and final step, shown in 2-19 c), was to synthesize intermediate networks for changes in pose. It must be noted that the synthesis of a new intermediate network, in this case, necessitated the warping and blending of *masks* in addition to images and flow. Warping and blending masks is similar to warping images, except that care must be taken to tailor the warping and blending algorithms to maintain the black-and-white nature of the mask pixels.

Experiments were performed with such a hierarchical eyes-mouth-pose network, and figure 2-20 shows two sequences of images generated from the same eyes-pose-mouth synthesis network. In the top row, the mouth is kept closed while the eyes and the pose is changed. In the bottom row, all three facial features are changed. In general, the new synthesis approach completely eliminated the interference problems described earlier. As a result of the fact that the intermediate images were generated using fewer neighbors, the synthesized images were also of better quality as well.

The 5-dimensional, 14-example network shown in figure 2-19 a) constitutes one of the networks that will be used in Chapter 3 to analyze novel image streams.

It is interesting to point out that the new synthesis approach is not a new paradigm at all, but a generalization. In the old synthesis method, *images* were warped and blended together to achieve novel, intermediate images. In the new method, this notion of warping and blending is extended to include not only *images*, but also *entire networks*. One can alternatively think of the synthesis technique as warping and blending *nodes*, where a node can be an image, a network, a network of networks, and so on.

As a consequence of this generalization, it is possible to scale up the eyes-pose-mouth network to include more pose dimensions. The difficulty in scaling up networks in this manner is that the cross-flows linking the different subnetworks will increase linearly with the number of *examples* within each subnetworks. To add another 7-example regional eyes-mouth subnetwork, for example, requires 7 new cross-flows linking the base subnetwork with the new subnetworks. The increase in the cross-flows presents a problem similar to the problem asociated with the increase in example images described earlier.

2.10 Discussion

In this section, we discuss some of the advantages and disadvantages of the image-based synthesis techniques described in this chapter.

Above all, the experiments with image-based synthesis networks have shown that they are capable of achieving an extremely high degree of realism, especially when compared to traditional computer graphics techniques in which 3-dimensional models

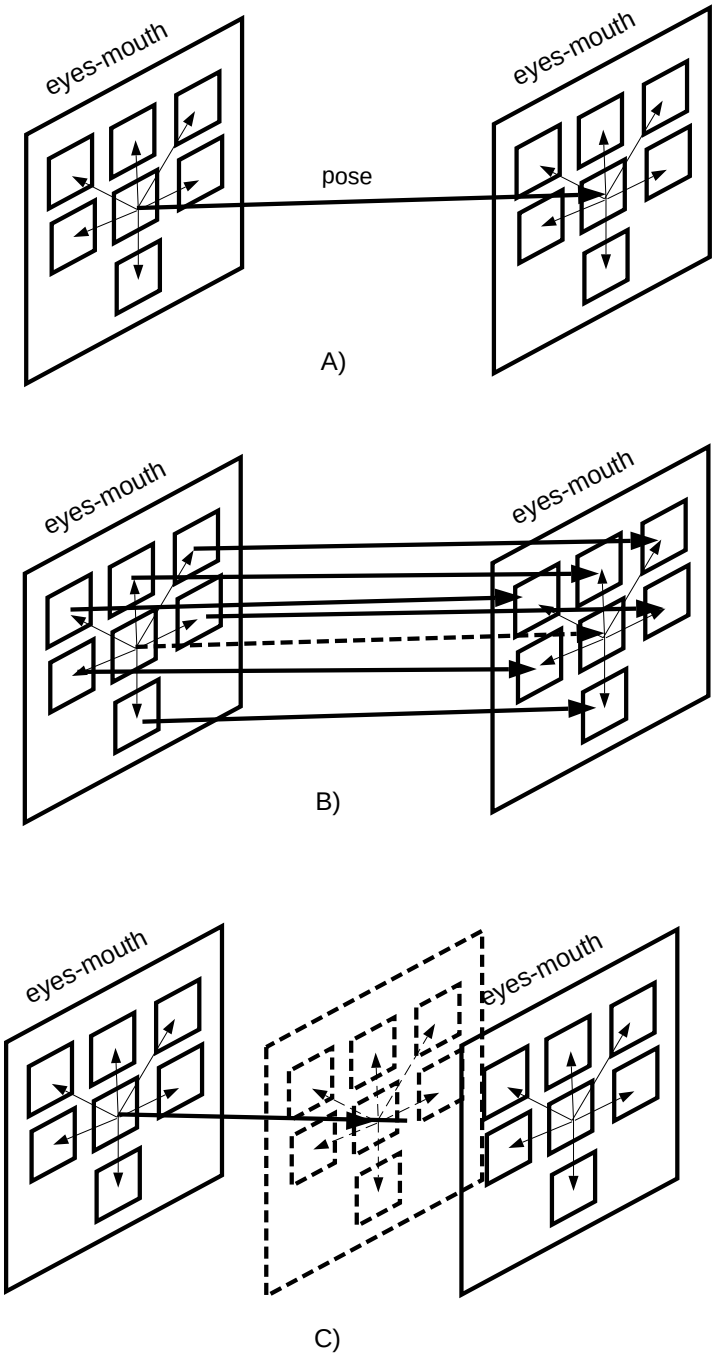


Figure 2-19: The stages of the new hierarchical synthesis approach for the 14-example, 5-dimensional, eyes-mouth-pose network.

Figure 2-20: Image sequences synthesized from the 14-example, 5-dimensional eyes-mouth-pose network

of the face are created. It quickly becomes apparent to many observers of these 3-dimensional models that, while many advances have been made in an effort to model human facial muscles, there is indeed a sense that the facial motions generated are not "realistic enough". The human perceptual system, it seems, is very sensitive to peculiarities in observed facial motions, especially around the region of the mouth and the eyes. Image-based methods, in capturing real facial imagery and real facial motion, are able to overcome many of these problems, sometimes to a frighteningly realistic degree.

It should be emphasized that the use of optical flow techniques is also a considerable advantage, since they enable one to capture realisitic human motion *automatically*, without the need to resort to any laborious manual specification of the correspondences, which many current, state-of-the-art morphing techniques usually require. Even more important than the fact that optical flow can yield correspondences automatically is the fact that composing flow across a series of consecutive, closely-spaced images enables one to obtain *good* correspondences.

It is also clear that combining the use of composed flow with a synthesis technique that involves warping and blending of images is also advantageous because, as has been shown, such a technique is capable of dealing effectively with facial movement that involves a large amount of occlusion, without the need to model visibility explicitly.

Of course, while image-based synthesis techniques provide a large amount of realism, they also suffer from a reduced degree of flexibility, in that one is always limited to traverse the space of images along the paths prescribed by the flow between them. In addition, the techniques for obtaining and constructing the synthesis networks are not completely automatic: the example-set designer must still choose the exam-

ple images, precompute the optical flow correspondences between the examples, and organize the examples and flows according to a particular parameter space.

As well, the image-based synthesis techniques are memory-hungry, requiring large amounts of RAM to store the images, the flows, and the cross-flows. The techniques are also slow, because the synthesis engine must perform many warps before it is able to produce the final synthesis output. Some success has been made at optimizing the synthesis routines for speed by reducing the quality of the final synthesized image. In certain applications, such as video email, time is not as important a factor as memory; in other cases, such as in the use of a synthesis network for virtual reality games, the ability to synthesize new images at interactive rates is more critical than the memory usage.

2.11 Summary

This chapter described the construction of the synthesis modules that are to be used in synthesis of facial image sequences from a set of high-level parameters. The synthesis network paradigm developed in Beymer, Shashua, and Poggio [5], was adopted because it is example-based, trainable, and can learn to generate novel views.

In specific, three networks were described:

- a network that synthesizes vertical and horizontal pose movements of the head, shown in figure 2-13,
- a network that synthesizes various mouth expressions involving significant occlusion, shown in figure 2-12,
- a network that synthesizes pose movements, eye movements, and mouth movements combined, diagramed in figure 2-19.

Chapter 3

Analysis

3.1 Overview

In this chapter, a model-based analysis algorithm is outlined which is capable of extracting a set of high-level parameters encoding head, mouth, and eye movements from novel image sequences. The models which the analysis algorithm employ are precisely the synthesis networks created in the previous chapter.

To be robust, the algorithm is designed to work given moderate changes in translation, onplane rotation, scale, lighting conditions, background changes, and hairstyle changes between the novel image sequence and the synthesis network used to extract the parameters. The particular cornerstones of the approach are:

- the technique *analyzes by synthesizing*. In other words, it uses the synthesis model itself to find the set of parameters that best fit the novel image.
- the technique is *flow based*, rather than image-based. At every iteration it tries to match a novel "flow" as opposed to a novel image. This allows the algorithm to be insensitive to lighting conditions.
- the technique is capable of extracting a set of affine parameters as well as the intrinsic set of parameters contained within the synthesis network. This allows the algorithm to be insensitive to changes in translation, scale, and onplane rotation.
- the technique analyzes only around the region of the head, because the synthesis networks are augmented with a segmentation scheme that allows the algorithm to seach only in regions around the head. This allows the algorithm to be insensitive to changes in background, and hairstyle.

After describing the details of the analysis algorithm, a series of experimental results are presented. The experiments involve testing the algorithm on a set of novel image sequences that involve changes in head position, scale, lighting conditions, hairstyle, mouth orientation, eye position, and pose. The tests are organized around the particular synthesis networks that are used by the algorithm as "models" of human facial motion. In specific, the tests are applied on:

- the two-dimensional, 9-example network that synthesizes vertical and horizontal pose movements of the head, shown in figure 2-13.

- the two-dimensional, 5-example network that synthesizes various mouth expressions involving signicant amounts of occlusion, shown in figure 2-12.
- the 14-example, 5-dimensional network that synthesizes pose movements, eye movements, and mouth movements combined, diagramed in figure 2-19.

3.2 Analysis Algorithm Features

3.2.1 Analysis by Synthesis

The analysis approach is, in fact, an *analysis-by-synthesis* approach, where the synthesis networks created in the previous chapter are themselves used for analysis. The most important reason for adopting this strategy is that it easily enforces on the analysis algorithm the constraint that the only parameters that may be extracted are those that are encoded by the synthesis model itself. In essence, the synthesis models contain the desired "prior information" which will guide the analysis algorithm in its effort to extract a set of parameters from a novel sequence.

3.2.2 Affine Parameters

Before analyzing with respect to novel image sequences, the synthesis networks developed in the first chapter must first be additionally parameterized with a set of affine parameters. This is needed because novel sequences usually involve movements of the head that are at scales, positions, and rotation angles that are different from those in the network, and, thus, any robust analysis algorithm needs to be able to "search" across a set of such scale, angles, and positions, to be able to "lock on" to the head in the novel image sequences. Of course, besides the fact that novel head movements may occur *at* different scales, rotations, and positions, it is also the case that novel head movements *involve* translation, scale, and rotation, which is another reason to incorporate affine parameters into the networks.

Augmenting the synthesis networks developed in chapter 2 with a set of four affine parameters (two translation parameters, an angle parameter, and a scale parameter), is straighforward. Essentially, the network first synthesizes the head at the intrinsic parameters constructed by the user, and then it performs an affine transformation according to the desired angle, position, and rotation.

Care needs to be paid to the rotation parameter, since the choice of the center of rotation can greatly affect the type of rotation obtained. We would ideally like to have the center of rotation coincide with the center of the head, but since we are not explicitly modelling any facial features (such as eye positions, head locations, etc.), the best approximation is to set the center of rotation to be the center of the image. This approximation is reasonable because, for the most part, the head will be located in the center of the image. In addition, since the analysis algorithm searches for the best set of affine parameters to match the synthesis network to the novel sequence, deviations in the center of rotations should not affect the ultimate outcome, but only prolong the search time.

Figure 3-1: An image that has undergone different affine transformations: horizontal translation, vertical translation, rotation, and scaling.

Figure 3-1 depicts an image obtained from an affine-augmented network that has undergone horizontal translation, vertical translation, rotation, and scaling, respectively. Of course, the affine transformations may also be composed.

3.2.3 Segmentation

In addition to augmenting the synthesis network with a set of affine parameters, it is also necessary to incorporate segmentation. This is needed because, in its effort to match the synthesis network with an incoming novel sequence, the analysis algorithm needs to match only on the region in the synthesized network that corresponds to the face. This will allow the algorithm to be less sensitive to background changes, as well as hairstyle and clothing changes.

In attempting to segment the head in a *network*, as opposed to segmenting the head in just an *image*, there is a need for a *flexible* segmentation scheme, because the outline of the head changes shape as the head changes pose, translates, rotates, and scales. One rigid mask is thus not capable of segmenting the head properly.

Consequently, a *network scheme* for flexible segmentation was adopted, where a network of the same dimensions and orientation as the corresponding image synthesis network is created, except that instead of images, the examples are masks. Each mask example serves to segment the head for the corresponding image example, and the correspondence flows relating the masks together are the same as those within the image synthesis network. The masks are defined by hand, although it is possible to use other automatic techniques. Whenever the synthesis network synthesizes a new image, it also synthesizes a new mask appropriate for the same image using the same warping and blending technique described in chapter 1, with some modifications. The hole-filling algorithm is modified to fill holes with "black" or "white" pixels, as opposed to filling a hole with a ramp of values defined by the endpoints of the hole. Also, the blending function, rather than blending the warped masks together, logically OR's all the masks together. In this manner, a pixel in the resulting synthesized mask is "on" if any of the warped masks have an "on" pixel at the correspoding point; else, the pixel is set to "off".

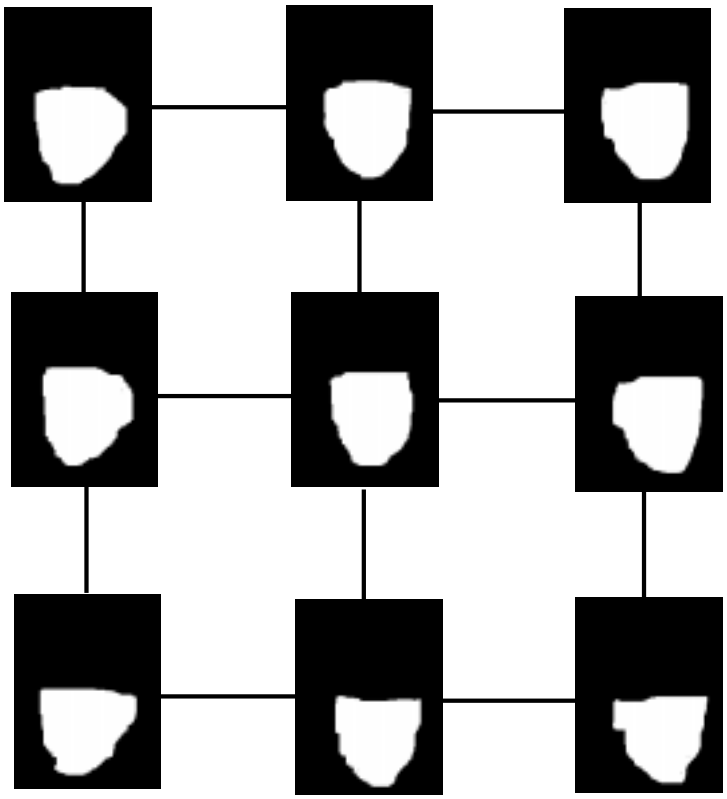


Figure 3-2: The masks associated with the 3-by-3 pose network.

Figure 3-2 depicts the masks that would be associated with the 3-by-3 pose network in figure 2-13. Figure 3-3 shows various segmented images, which are synthesized from the 3-by-3 pose network of figure 2-13 augmented with affine parameter perturbation and the segmentation masks.

It should be noted that, for the purposes of analysis, we have modified our concept of a synthesis network to incorporate different *layers*, to borrow the term from the work of Wang and Adelson [26]. One layer of the network thus consists of images, while a second layer consists of masks.

3.2.4 A Flow-Based Error Metric

A key feature of our analysis algorithm is that instead of using the embedded synthesis network to synthesize *images* to match to the incoming novel images, and thereby have to rely on an *image-based* error metric, the algorithm instead tries to match the incoming novel *flow*. For every iteration, the algorithm computes the optical flow between two consecutive incoming novel frames, and then attempts to find the best matching flow from within its embedded synthesis network.

The rationale for using a flow-based metric, as opposed to an image-based metric, is that trying to minimize a flow-based error metric is probably less susceptible to

Figure 3-3: Various segmented, affine-perturbed, synthesized images generated from the 3-by-3 pose network.

noise and local minima than trying to minimize an image-based metric. In addition, trying to match incoming novel flows as opposed to incoming novel images allows the analysis algorithm to be more invariant to lighting changes.

3.2.5 Parameter Perturbation Strategy

The analysis-by-synthesis algorithm is based on *iterative, local, independent perturbations of the synthesis parameters*. The steps of the algorithm are as follows:

1. For a novel flow obtained from two consecutive novel images (say images A and B) in the sequence, the parameters of the embedded synthesis model are perturbed. The perturbations include the affine parameters, and vary each parameter independently in the positive and negative directions by a small *delta* factor.

2. For each set of perturbed parameters obtained, the algorithm then synthesizes a flow from the network that corresponds to the perturbation.

3. The algorithm then computes the Euclidean distance between each perturbed flow and the novel flow, and finds the closest synthesized flow of the set.

4. The algorithm then repeats steps 1 throught 3 above, iteratively perturbing the set of parameters associated with the closest synthesized flow found in step 2.

5. For each iteration in step 4, the synthesized flow that yielded the overall smallest distance with respect to the novel flow is preserved; if a set of perturbations do not yield any new synthesized flows that reduce the overall minimum, the delta factors are halved and the iterations proceed once again. Thus when the algorithm gets close to the optimum synthesized flow, it proceeds with smaller and smaller perturbations to achieve a better match. The iterations terminate when the delta factors have been halved to a degree where perturbations made using those factors do not make any significant changes in the synthesized flows.

6. Once a parameter estimate is obtained for the given novel flow, the algorithm computes the next consecutive incoming novel flow in the sequence (say, between images B and C), and starts to perturb around the set of parameters found in the previous iteration. This whole process is performed across the entire sequence.

Figures 3-4 depicts a hypothetical set of iterations of the algorithm where the embedded synthesis network containes 4 examples in 2 dimensions. Because the network only has two intrinsic dimensions, every iteration generates a set of only four perturbed flows (two for each intrinsic dimension), except in cases where the perturbations would lead to points that lie outside the designated range of the example set. Ordinarily there would be an even larger number of perturbations due to the addition of the affine parameters, but these are not shown in the figure.

The novel flow that the perturbed flows are compared to is shown in figure 3-4 as the dark, thick arrow. The small round circle in each iteration indicates the particular flow in the *previous* iteration that yielded the closest match to the novel flow. Consequently, all the perturbations for the current iteration occur *about the small round circle point*. As the iterations progress, and the match comes closer to the novel flow, the perturbations become smaller, as may be seen in the figure.

3.2.6 Synthesizing Flow for Analysis

Step 2 from the previous section needs further elaboration. There are two main methods to obtain, for a set of perturbed parameters, the corresponding synthesized flow: the first, *direct* method is to use equation 2.14, and synthesize the flow y for the set of perturbed parameters x. The other, *indirect* method is to synthesize the perturbed image first, and then compute optical flow between it and the reference image.

Both methods are depicted in figure 3-5; the desired perturbed flows are the dark, thick arrows. In the direct method, the flows produced by equation 2.14 originate from the reference image of the network. Consequently, the synthesized flows must be subtracted from each other to yield the desired perturbed flows. In the second method, the images are synthesized first, and then the desired perturbed flows are computed using direct application of the optical flow algorithms between the images.

The two techniques are not equivalent. The second is much more computationally intensive, since an image must be rendered and then flow computed. However, it yields much better flow estimates than the first method. The key to the benefit of

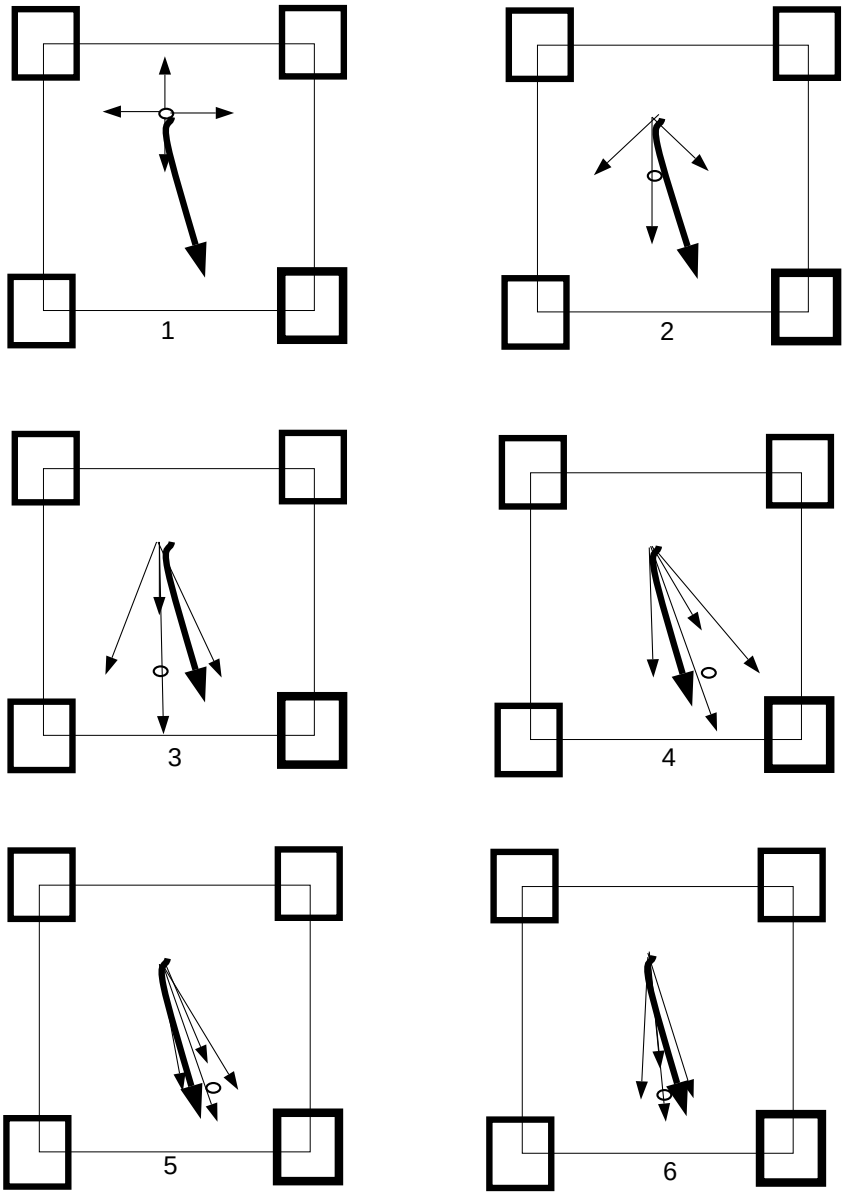


Figure 3-4: A series of perturbations that the analysis algorithm performs to match a novel flow, depicted as the solid black arrow. The small round circle in each iteration indicates the perturbed flow in the previous iteration that yielded the closest match. Notice that, as the iterations progress, the perturbations become smaller and smaller.

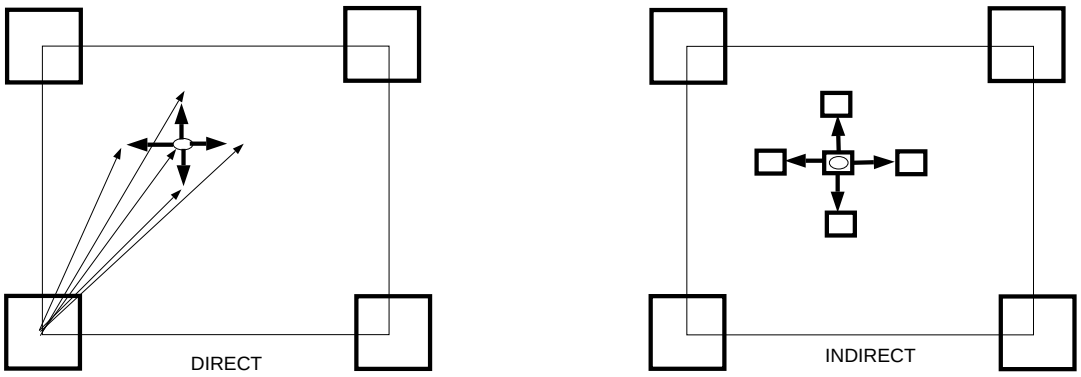


Figure 3-5: Direct and indirect methods to obtain the perturbed flows, shown as the thick black arrows in both diagrams. The direct method synthesizes flows normally, and subtracts them to obtain the desired perturbed flows. The indirect method synthesizes images, and uses optical flow to obtain the desired perturbed flows.

the indirect method lies in the fact that synthesis involves a combination of warping and blending. As was described in chapter 2, it is the combination of both warping and blending that enables the synthesis network to produce correct novel images despite the shortcomings of the correspondences computed by the optical flow algorithms. The first, direct method produces a set of flows that are based *entirely* on the optical flow algorithms themselves, and, as such, will suffer from linearization errors. The second method, on the other hand, first synthesizes images, thus overcoming the errors in correspondence, and then computes optical flow. *Since the perturbations are small, the images will be close, and the resulting flows will not suffer from the linearization errors.*

It is also advantageous to use the second method because the synthesis equation 2.14 does not yield the flows associated with the affine perturbations. Synthesizing the affine-perturbed images and then computing flow, however, allows such flows to be computed.

3.2.7 The First Image in a Sequence

The first image in the novel incoming sequence needs to be treated differently from the other images. For this special case, we would still like to be able to extract its postion in parameter space using a flow-based metric rather than an image-based metric, but there is no prior flow within the sequence itself against which to match. Consequently, the algorithm computes the flow from the reference image in the network to the first novel image, and then applies the iterative parameter perturbation technique to find the closest synthesized flow. This strategy suffers from the weakness that if the optical flow fails due to the fact that the heads are too far away from each other, then the extracted parameters for the first image will be incorrect. Consequently, in the novel sequences that we used to test the analysis algorithm on, the head in the initial frame

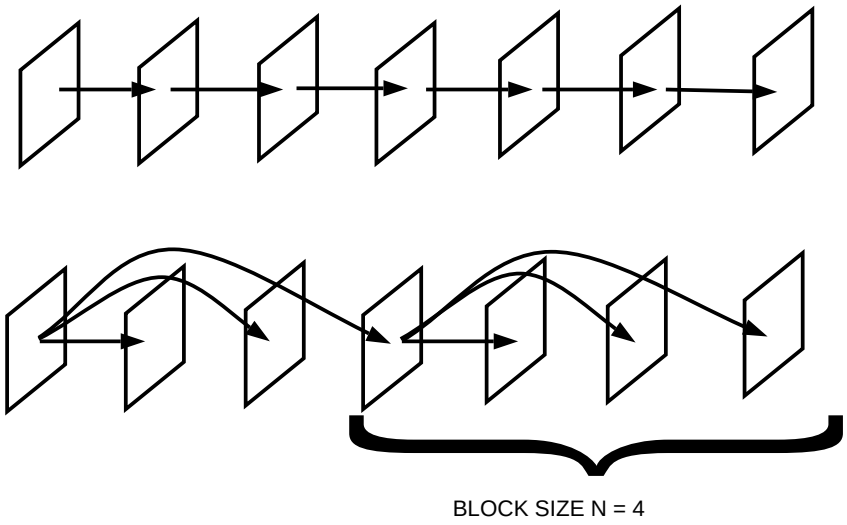


Figure 3-6: Two strategies for traversing the novel incoming image sequence. In the top, the flow between consecutive images is matched by the synthesis model. In the bottom, the flow between the head of the block and the current image is matched.

was not placed too far away from the head in the reference image of the embedded synthesis network, although, of course, significant deviations in translation, rotation, scale, pose, and other variables did exist nevertheless.

3.2.8 The Rest of the Sequence

After the parameters for the first image (image A) are determined, the algorithm computes optical flow between images A and B, and tries to match that flow by perturbing the parameters of the first image. This process is subsequently iterated for the novel flow between images B and C, C and D, and so on.

However, because the frame rate at which the novel sequences were captured was high (about 20 frames per second), the flow obtained between two consecutive novel images tends to be too small and insignificant for the network to match accurately, so the following slightly modified technique was used: the novel incoming image sequence is divided into *block sizes* of N frames each. For each iteration i of the algorithm within a particular block, the flow between the head of the block and the ith frame is matched. The iterations proceed until $i = N$, at which point the Nth image is chosen as head of the next block, and the whole process repeats.

Figure 3-6 depicts the two different strategies. The block-based strategy allows more significant flows to be obtained, since more time will have elapsed between the head of the block and the ith image in the block as i gets closer to N. Of course, there is also a need to periodically start over with a new block so that the optical flow routines do not attempt to compute flow between images that are too far apart.

Since many of our test examples were short, the block size was usually the number

of total frames in the sequence.

3.2.9 Resolving Translation-Pose Confusion

One of the first problems that emerges with the analysis algorithm as described so far is that the flows associated with translation are very similar to the flows associated with pose movement. Consequently, the algorithm sometimes chooses a translation perturbation instead of the more correct pose perturbation, simply because the first perturbation minimized the overall distance to the novel flow more than the the latter perturbation did.

To alleviate problems associated with the similarity in the flow fields between pose movements and translation movements, *the analysis search is split into two stages*: a stage in which the synthesis network's intrinsic parameters, such as pose, mouth orientation, etc., are perturbed, and a stage in which the affine parameters are perturbed. For the first image, the analysis algorithm first perturbs the affine parameters, to move the head close to its position in the first image, and then it perturbs the intrinsic parameters to obtain the correct pose and mouth orientation. For the subsequent images in the sequence, the order is reversed: first the intrinsic parameters are perturbed, then the affine parameters, followed by a final intrinsic perturbation. The rationale behind that strategy is that as the head undergoes an intrinsic change between images, the first intrinsic set of perturbations will be able to match the flow correctly; if the head instead undergoes an affine change, the second, affine set of perturbations will be able to yield the correct match; if the head undergoes an intrinsic *and* an affine change between images, a third and final intrinsic perturbation stage is needed for some added precision after the affine perturbation stage.

3.3 Experiments

In this section, the results from a series of analysis experiments are presented. The experiments are organized according to the particular embedded synthesis network used for analysis of novel image sequences. In particular, we present:

- analysis experiments performed using the two-dimensional, 9-example network that synthesizes vertical and horizontal pose movements of the head, shown in figure 2-13.
- analysis experiments performed using the two-dimensional, 5-example network that synthesizes various smile/open-mouth expressions, shown in figure 2-12.
- analysis experiments performed using the 14-example, 5-dimensional network that synthesizes pose movements, eye movements, and mouth movements combined, diagramed in figure 2-19.

To test the robustness of the analysis algorithm, the novel image sequences included moderate differences in facial location, scale, and angle, in addition to lighting

differences. *Also note that the author resolved to shave his head to test the algorithm's robustness with respect to changes in hairstyle!*

The analysis parameters that were extracted from the novel sequences were subsequently used to resynthesize the sequence, and hence complete the model-based analysis-synthesis loop that was shown in figure 1-1. Resynthesis is useful to gauge the quality of the extracted parameters, although a static medium such as this thesis is not capable of suitably conveying the dynamics of the synthesized motion.

In the results presented in the following sections, the novel sequence will be juxtaposed against the synthesized sequence, followed by a plot of the extracted analysis parameters.

3.3.1 Pose Experiments

Four experiments were performed involving the 3-by-3 pose network shown in figure 2-13, and the results are shown in figures 3-7, 3-9, 3-11, and 3-13.

The results obtained showed that the analysis algorithm is capable of extracting pose parameters reasonably well, and that splitting the parameter search into separate intrinsic and affine searches did indeed resolve much of the pose-translation confusion that had been observed in preliminary experiments.

3.3.2 Mouth Experiments

Only one analysis experiment was performed to test the ability of the analysis algorithm to capture movements of the mouth. The embedded synthesis network used was the 2-dimensional, 5-example network that synthesized various smile/open-mouth expressions, shown in figure 2-12, and the result is shown in 3-15.

Unfortunately, in this case the model was created with the head at a lower scale than the head in the novel image sequences. Consequently, the analysis algorithm was not able to match the novel mouth movements to a sufficient *fidelity*, and thus, there is a need for a better mouth model at a higher resolution. Nevertheless, the synthesized mouth movements did match the novel mouth movements *to the best of the model's ability.*

In addition, analysis of mouth movements proved to require modification of the initial delta factors of the algorithm, which determine the size of the initial perturbations. The first set of delta perturbations were too small, and the algorithm terminated at an incorrect local minimum. Enlarging the delta factors improved the final results considerably.

It should also be noted that the presence of the cap did not affect the analysis.

3.3.3 Eyes, Pose, Mouth Experiments

Three analysis experiments were performed to test the ability of the analysis algorithm to capture movements of the eyes, mouth, and pose combined. The embedded synthesis network used was the 5-dimensional, 14-example network that synthesized various eye positions, mouth movements, and pose movements combined, diagramed

in figure 2-19, and the results are shown in 3-17, 3-19, and 3-21. The first test sequence involved mouth movement alone, the second involved only eye movements, and the third involved combined mouth, eyes, and pose movements.

The results in general were good for all three sequences, although in the third sequence the synthesized eye movements did not match the novel ones very well. Also, the mouth in the second and the third sequences did not open to the complete extent that it did in the novel sequence.

3.3.4 Affine Experiments

Four analysis experiments were performed to test the ability of the analysis algorithm to capture movements of the head that are largely affine movements. The embedded synthesis network could have in principle been any of the networks used in earlier experiments, but in this case the two-dimensional, 5-example network that synthesized various smile/open-mouth expressions, shown in figure 2-12, was used, and the results are shown in 3-23, 3-25, 3-27, and 3-29,

In general, the analysis algorithm worked well in the case of affine movement, although the printed results in this thesis do not adequately convey that, due to their static nature.

3.4 Discussion

In general, the preliminary experiments performed are heartening, but more tests are needed to test the algorithm more thoroughly, as well as to address out some of its problems, which include:

1. It currently takes a long time to analyze a sequence a novel image. The time complexity of the algorithm depends on several factors:

   - The time it takes to synthesize an image from the synthesis network, which varies with the complexity of the type of network.
   - The number of parameters there are in the model.
   - The time it takes to compute flow.

   No formal timing tests were made, but it took on average about several hours for a simple network to analyze a novel image sequence of about 30 frames, and half a day to a day for some of the complicated networks to analyze a novel sequence of the same length. Possible changes and additions to improve the time complexity of the analysis algorithm run the gamut from caching the perturbed flows to avoid recomputation if the parameters are revisited, to possible implementation of the synthesis routines in hardware. However, improvements to the algorithm were beyond the scope of this thesis.

2. Clearly, since the metric used to match the synthesized images to the novel images is flow-based, the algorithm is extremely sensitive to situations in which the optical flow algorithms fail. This may arise in situations where the images are too far apart or too dissimilar. In general, more robust correspondence algorithms are needed, but this too was out of the scope of this thesis.

3. Because the algorithm searches across the set of parameters that minimizes the Euclidean distance between a novel flow and a synthesized flow, the algorithm can fall into local minima. Local minima may be avoided with larger initial perturbations.

3.5 Applications and Further Work

On the positive side, the example-based synthesis and analysis paradigm adopted in this thesis seems general enough to be possibly useful in many application scenarios. For example, the synthesis paradigm by itself may be used to create photorealistic human avatars for use in virtual chat rooms and virtual reality games. The analysis paradigm by itself may be useful for speech analysis and eye tracking of human subjects.

Of course, the main application that motivated the adoption of the approaches used in this thesis is *model-based coding*, whose goal it is to compress image sequences by extracting high-level parameters from the images. In principle, as shown in this chapter, each image may be compressed to a few bytes encoding the face's location, mouth orientation, and eye position. Such ratios far exceed those of traditional hybrid interframe coding algorithms such as MPEG, which achieves an average compression ratio of about 30:1. Of course, the disadvantage of model-based coding schemes is that they can only code the particular modelled objects, and can only extract the particular parameters built into the models themselves.

The use of the model-based analysis and synthesis paradigm described in this thesis for applications such as video email and video teleconferencing will require solving another key problem that has not been discussed in this thesis at all: how to ensure that the synthesis network at the receiving end reconstructs the face image sequence *with the original texture*. As can be seen in the results of this chapter, models with facial images taken at different periods of time will lead to reconstructed image sequences that look completely different from the original novel sequence.

There are two possible ways to solve the *texture* problem: the first technique, discussed in [21], is to use traditional DCT-based coding techniques to code the *residual error* between the synthesized image and the novel image, and then send the coded residuals over to the receiving end along with the analyzed parameters. The synthesis network at the receiving end synthesizes the new image using the extracted analysis parameters, and then adds the decoded residuals to achieve the correct texture. Of course, the idea behind this approach is that the coded residuals will occupy fewer bits than coding the entire image by itself, but this remains unclear until experiments are made.

The other approach is use the extracted parameters, at the analysis end, to sample the incoming image stream in a much more intelligent fashion. For example, if an extracted parameter streams exhibit linear behavior, one can send to the receiving end only two images and the optical flow between the two images, and instruct the synthesis network at the receiver to morph linearly between the two images. Of course, in the case of multiple parameter streams, the sampling algorithm must look at all the streams simultaneously, but, in general, it is easier to look at such one-dimensional parameter streams than at motion fields and images themselves. In an MPEG context, this approach may be viewed as employing the extracted model-based parameters to sample I-frames more intelligently. The texture problem is solved because the original frames are sent to the receiving end instead of only the parameters.

In both approaches described above, of course, the compression ratios will decrease because more information is being sent to the receiving end. The second approach, in particular, may lead to lower compression ratios when compared to MPEG, because the original frames sent over to the receiver will not be intra-coded, as in MPEG. For short sequences, where there may be a large number of sampled frames sent when compared to the length of the sequence, MPEG would probably be a better algorithm to use. For longer sequences, however, our model-based approach will win out because fewer original frames will be sent in contrast to MPEG, which blindly transmits an I-frame at regular intervals.

Of course, much further work still needs to be done before model-based algorithms become a reality. In particular, besides improvements and enhancements to the analysis algorithm itself, there is also a lot of room to explore the creation of more complex synthesis models. Possibilities include extending the network that synthesizes *limited* pose movement, eye movement, and mouth movement combined to *full* head pose movement. Also, the network that synthesizes various mouth expressions may be extended to include a set of visual speech visemes. Instead of human heads, networks for full human bodies may be attempted. Finally, the synthesis paradigm may be applied to rigid objects instead of semi-rigid ones such as a human head.

3.6 Summary

In this chapter 3 synthesis models constructed in chapter 2 were used in experiments in which novel image sequences were analyzed using an iterative, local, independent parameter perturbation strategy. In specific,

- four experiments were performed to test the analysis of pose movements with the two-dimensional, 9-example network shown in figure 2-13.
- one experiment was performed to test the analysis of mouth expressions with the two-dimensional, 5-example network shown in figure 2-12.
- three experiments were performed to test the analysis of mouth, pose, and eye movements with the 14-example, 5-dimensional network diagramed in figure 2-19.

- four experiments were performed to test the analysis of affine movements of the head with the two-dimensional, 5-example network shown in figure 2-12.

In general, the results of the experiments were very heartening, and suggest that the analysis algorithm may be suitable to analyze a wide array of human facial movements. Further tests are needed, however, to determine strategies to improve the algorithm's fidelity (sensitivity to small, detailed movements).

Figure 3-7: A novel sequence with leftwards pose movement, juxtaposed along with the synthesized sequence.

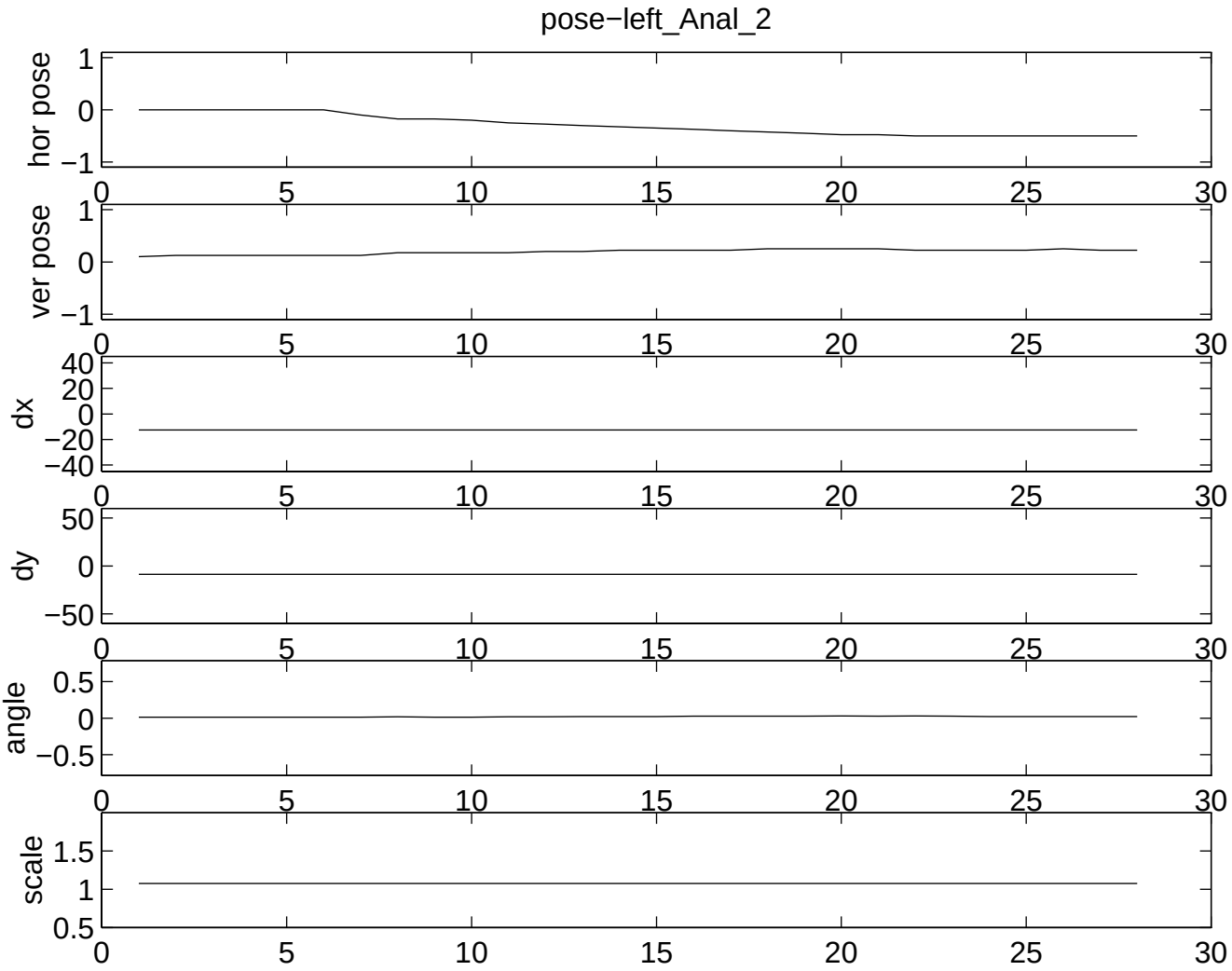


Figure 3-8: The parameters extracted from the previous novel sequence.

Figure 3-9: A novel sequence with rightwards pose movement, juxtaposed along with the synthesized sequence.

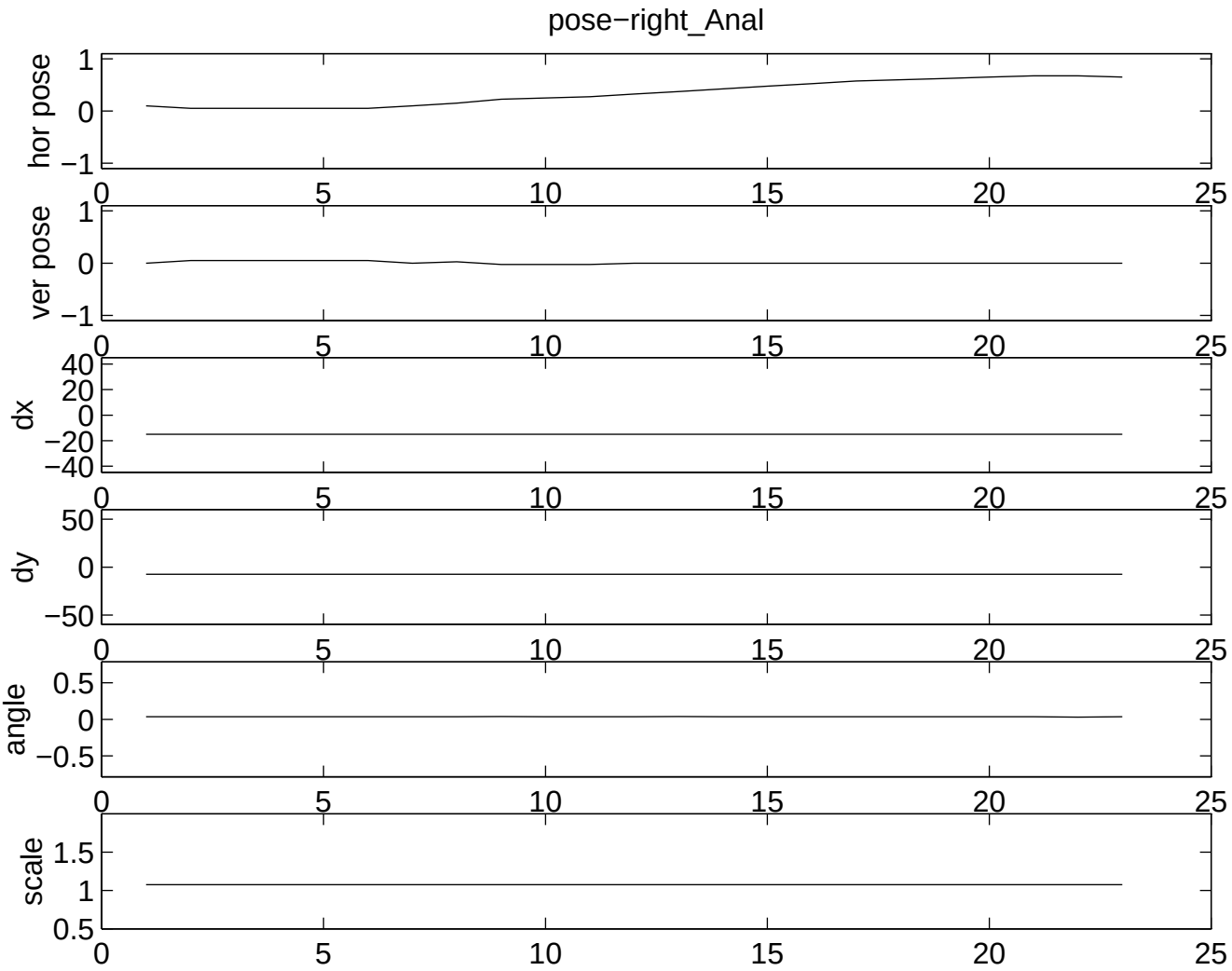


Figure 3-10: The parameters extracted from the previous novel sequence.

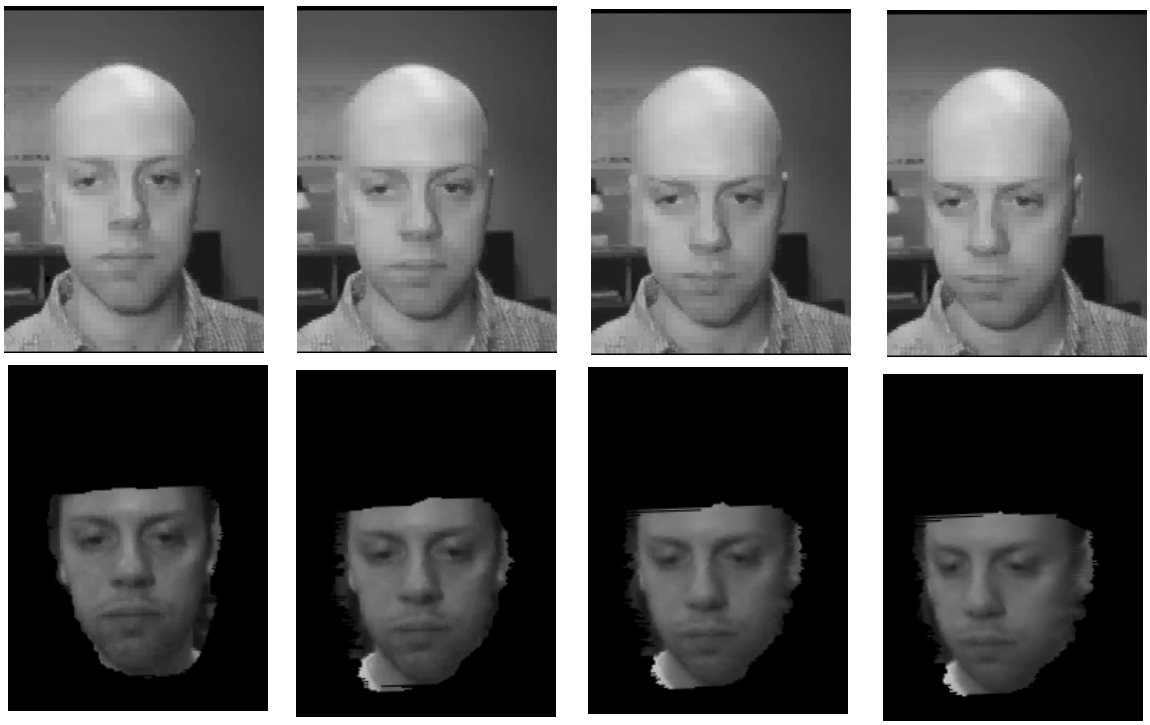


Figure 3-11: A novel sequence with bottom-left pose movement, juxtaposed along with the synthesized sequence.

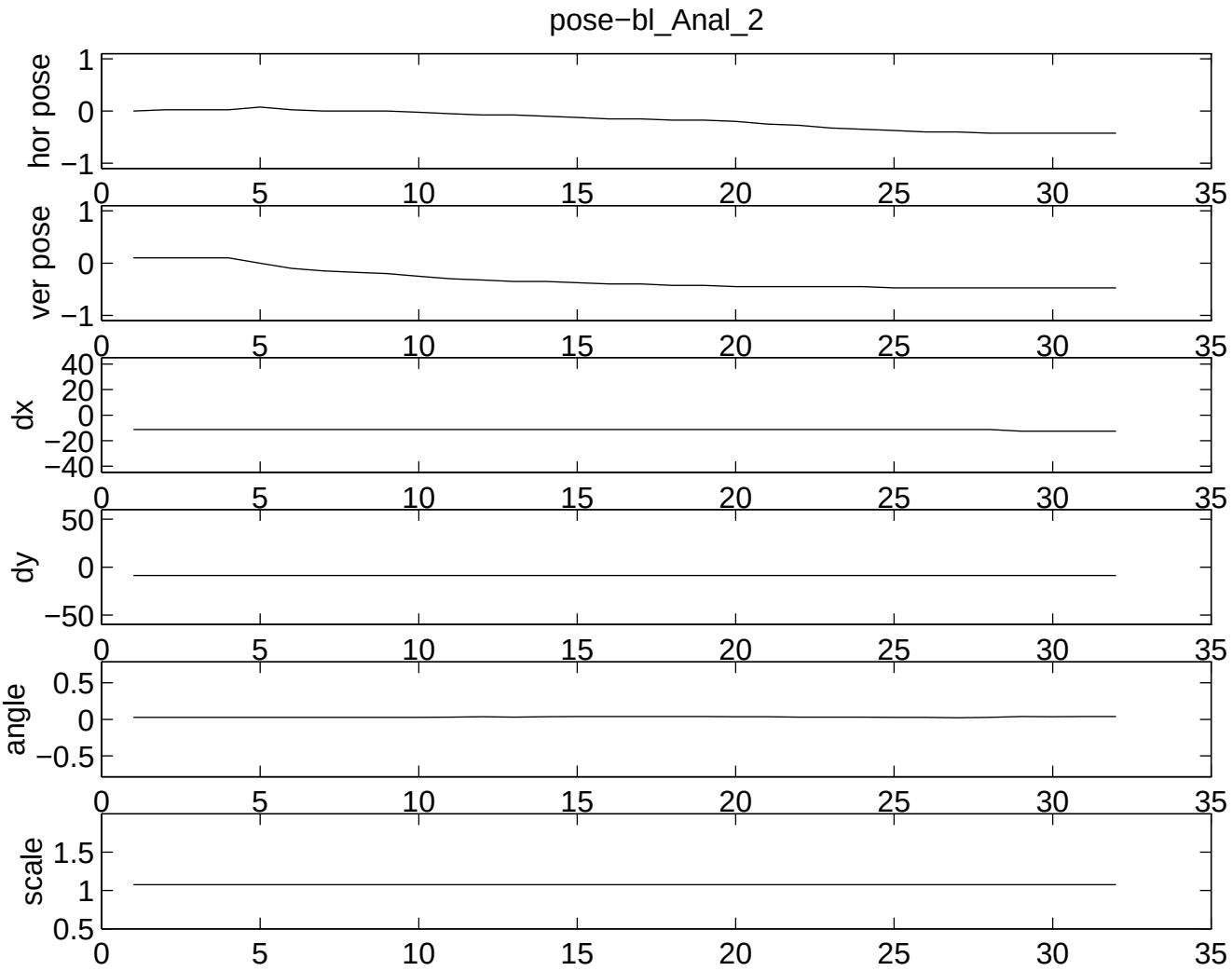


Figure 3-12: The parameters extracted from the previous novel sequence.

Figure 3-13: A novel sequence with top-right pose movement, juxtaposed along with the synthesized sequence.

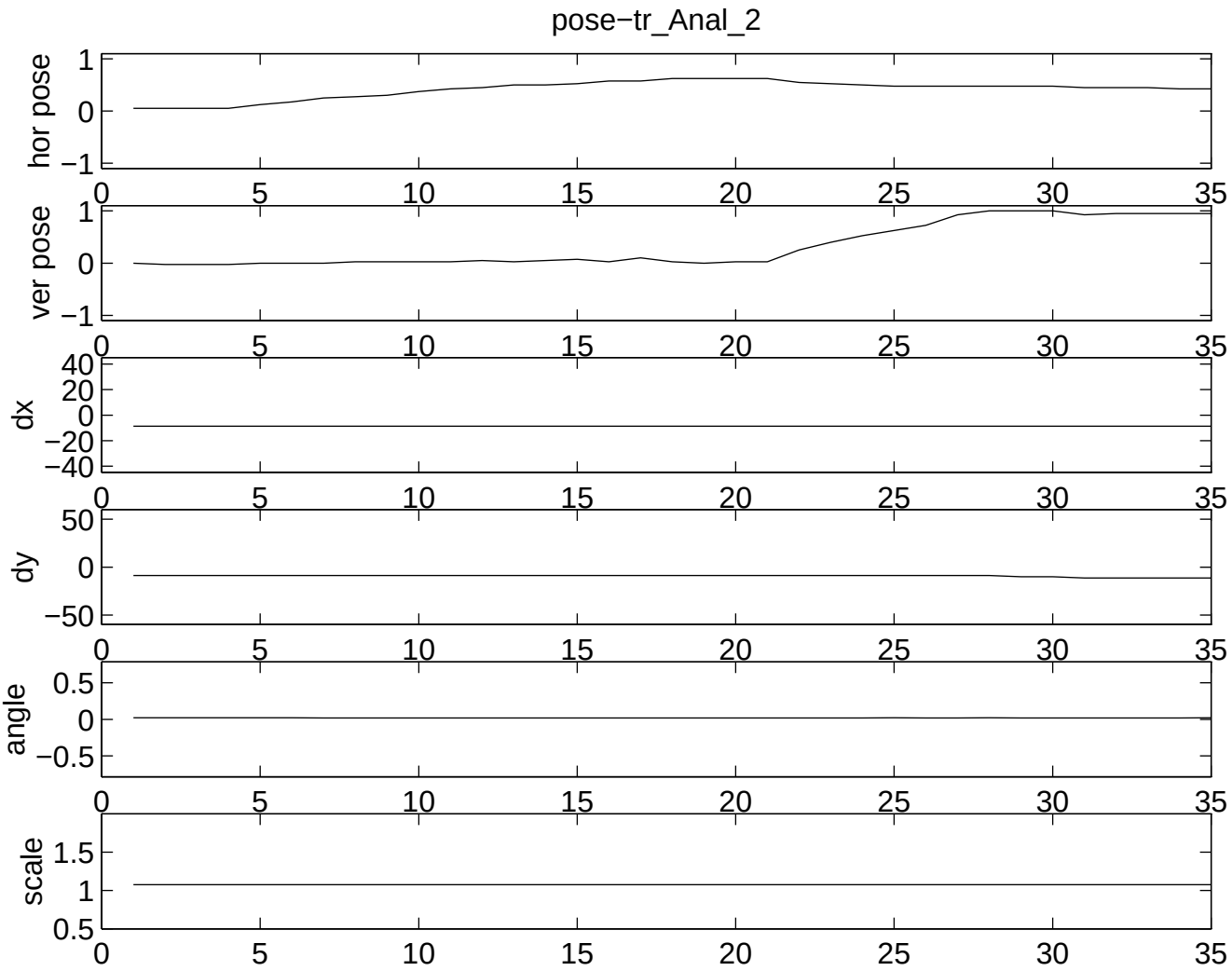


Figure 3-14: The parameters extracted from the previous novel sequence.

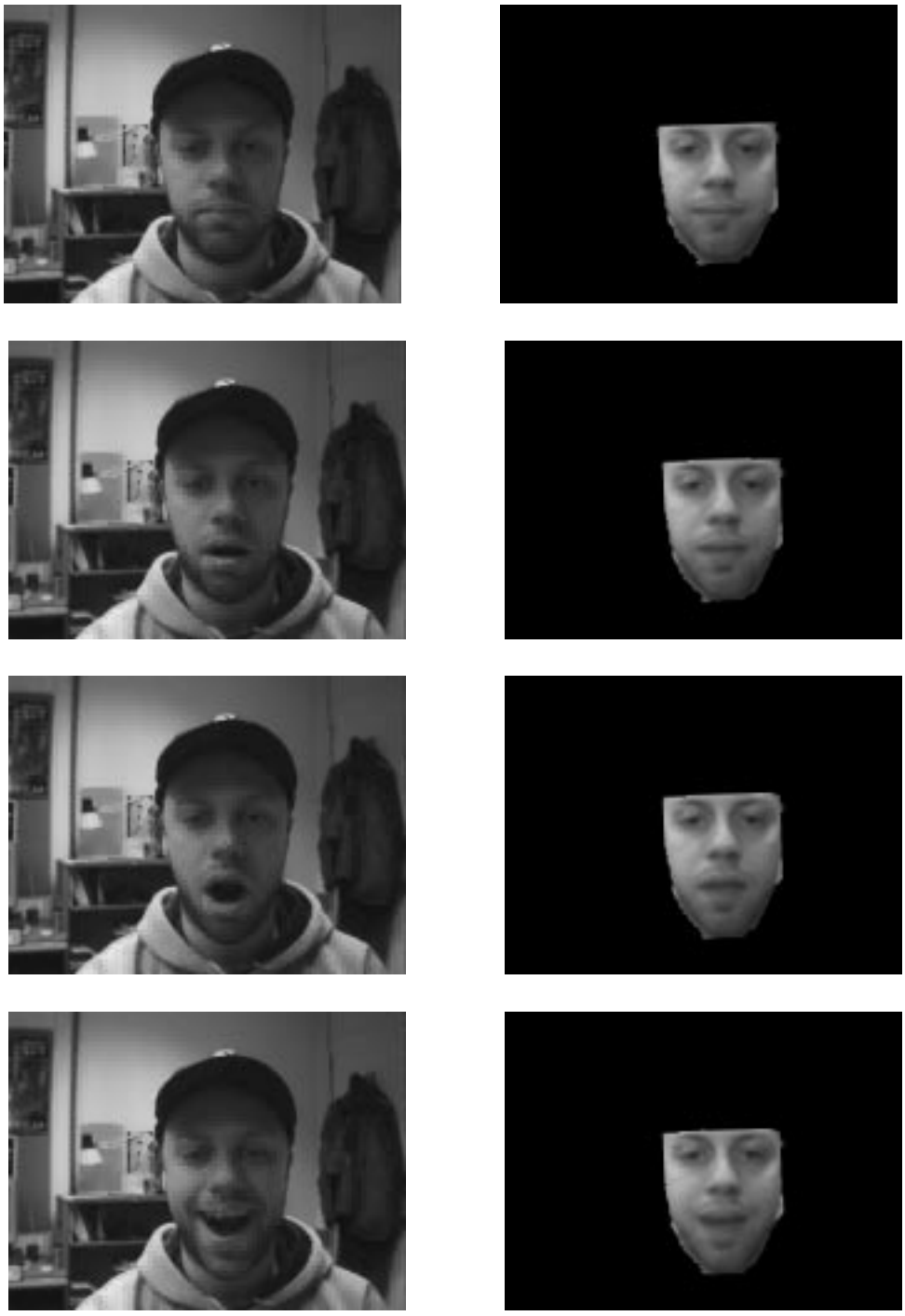


Figure 3-15: A novel sequence with mouth movement, juxtaposed along with the synthesized sequence.

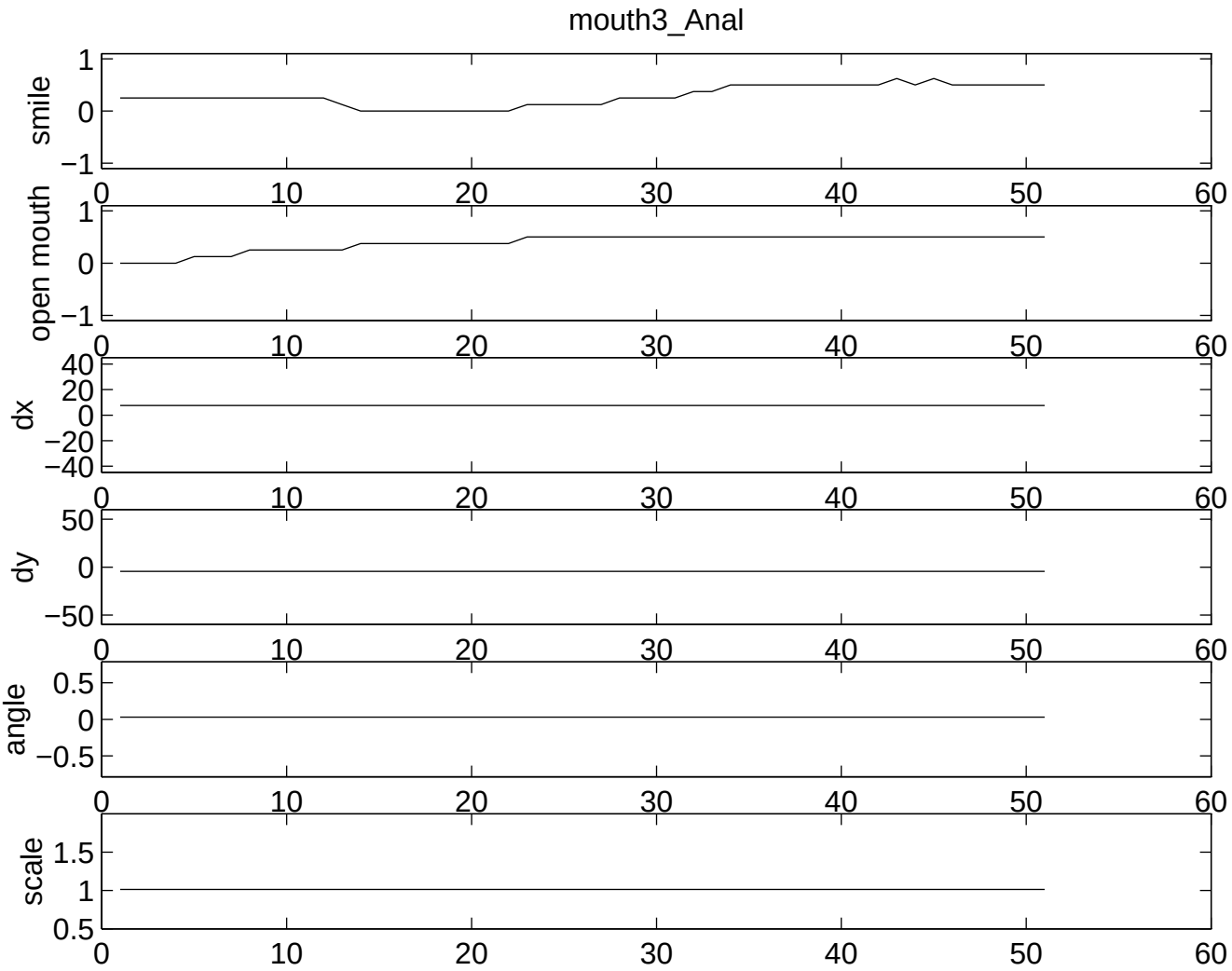


Figure 3-16: The parameters extracted from the previous novel sequence.

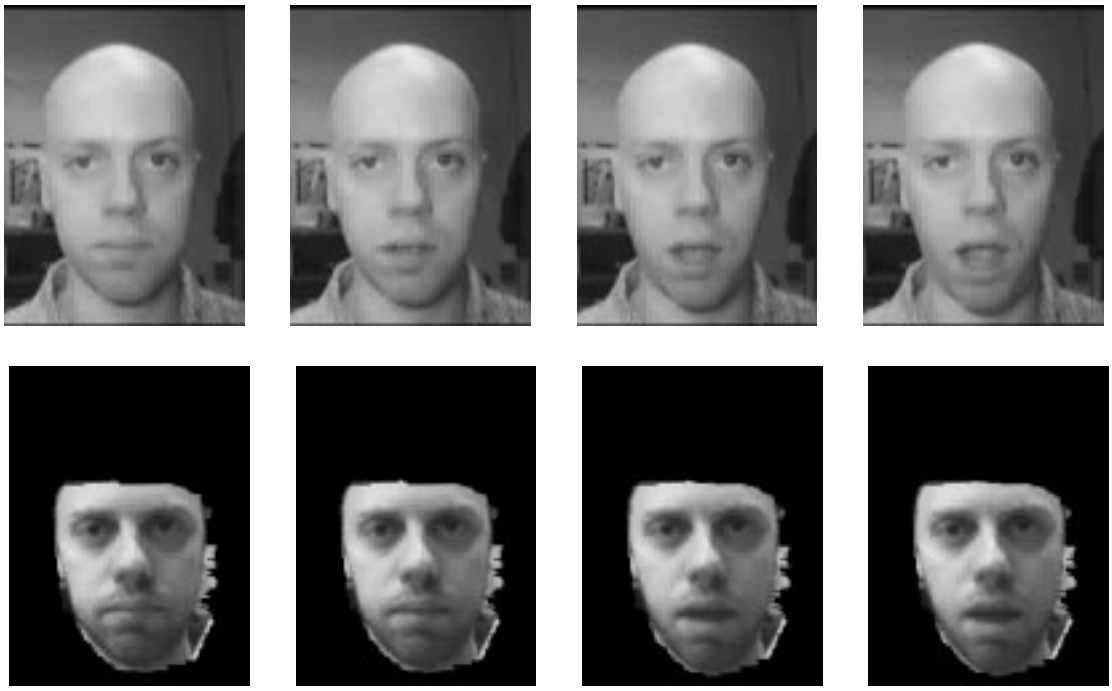


Figure 3-17: A novel sequence with mouth movement, juxtaposed along with the synthesized sequence.

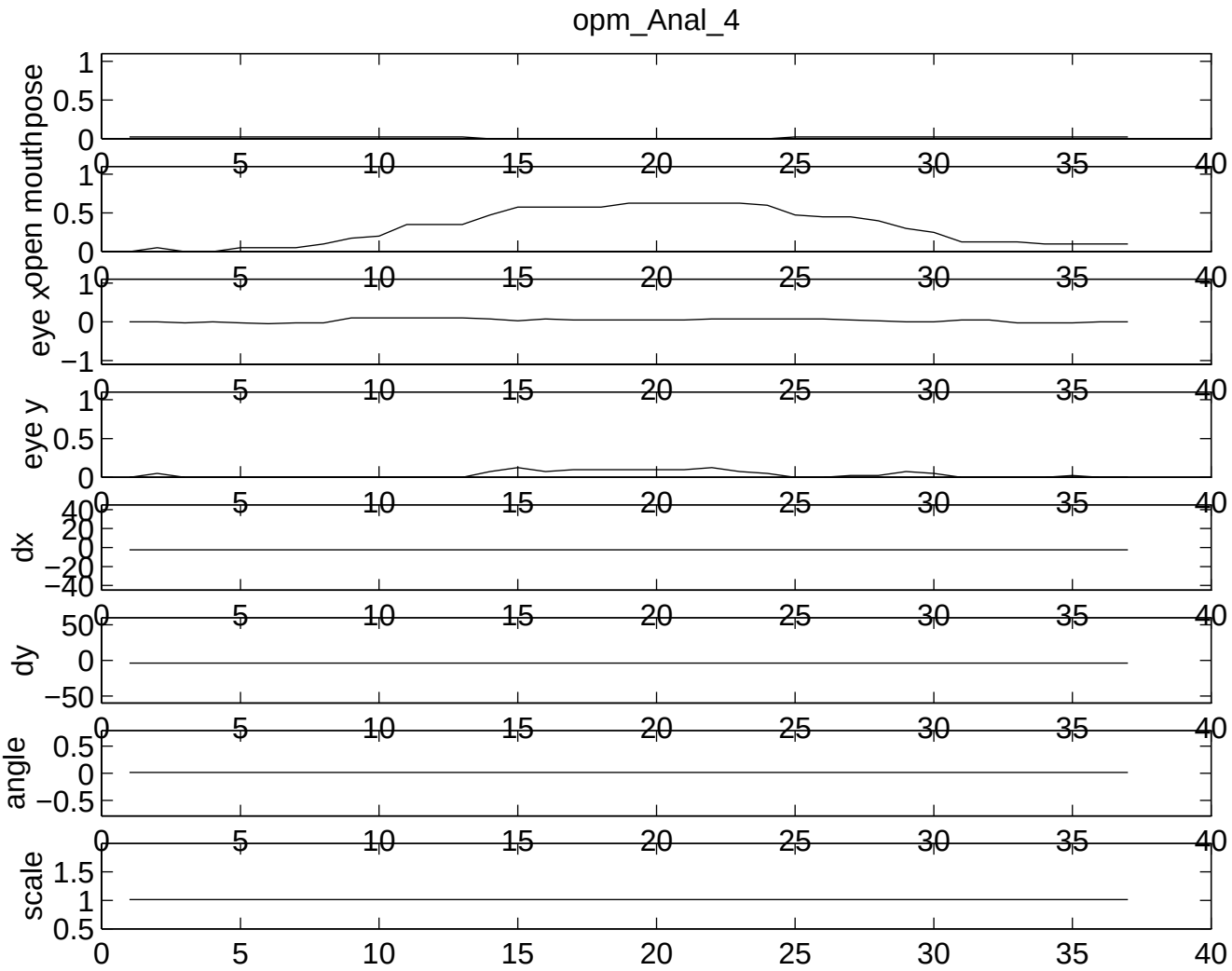


Figure 3-18: The parameters extracted from the previous novel sequence.

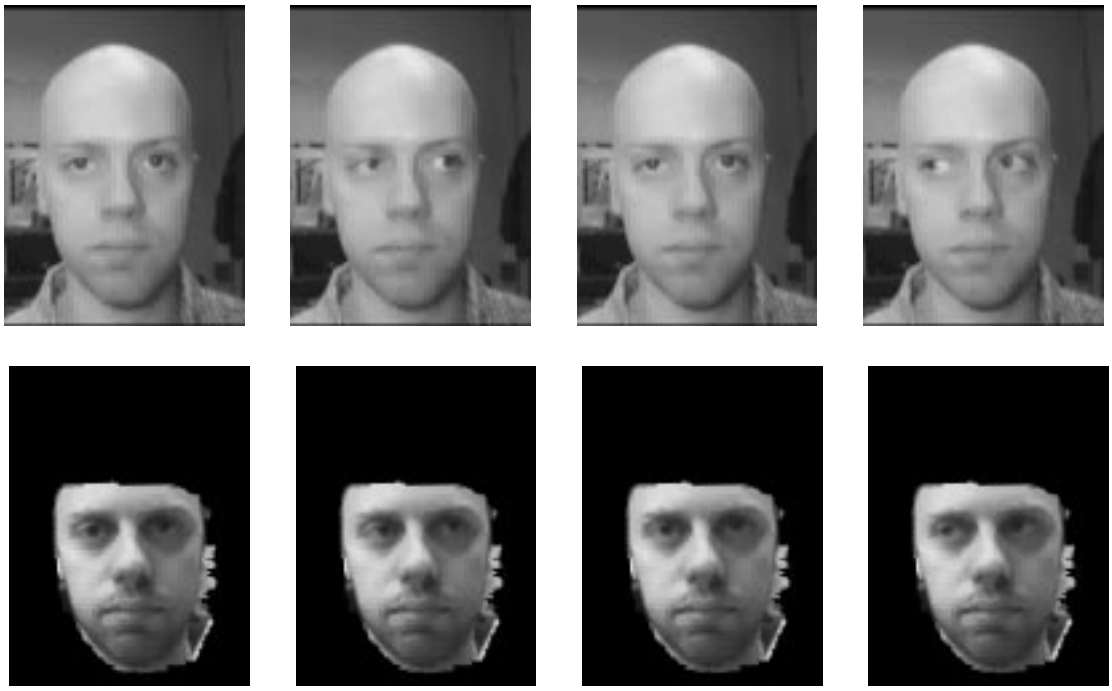


Figure 3-19: A novel sequence with eye movement, juxtaposed along with the synthesized sequence.

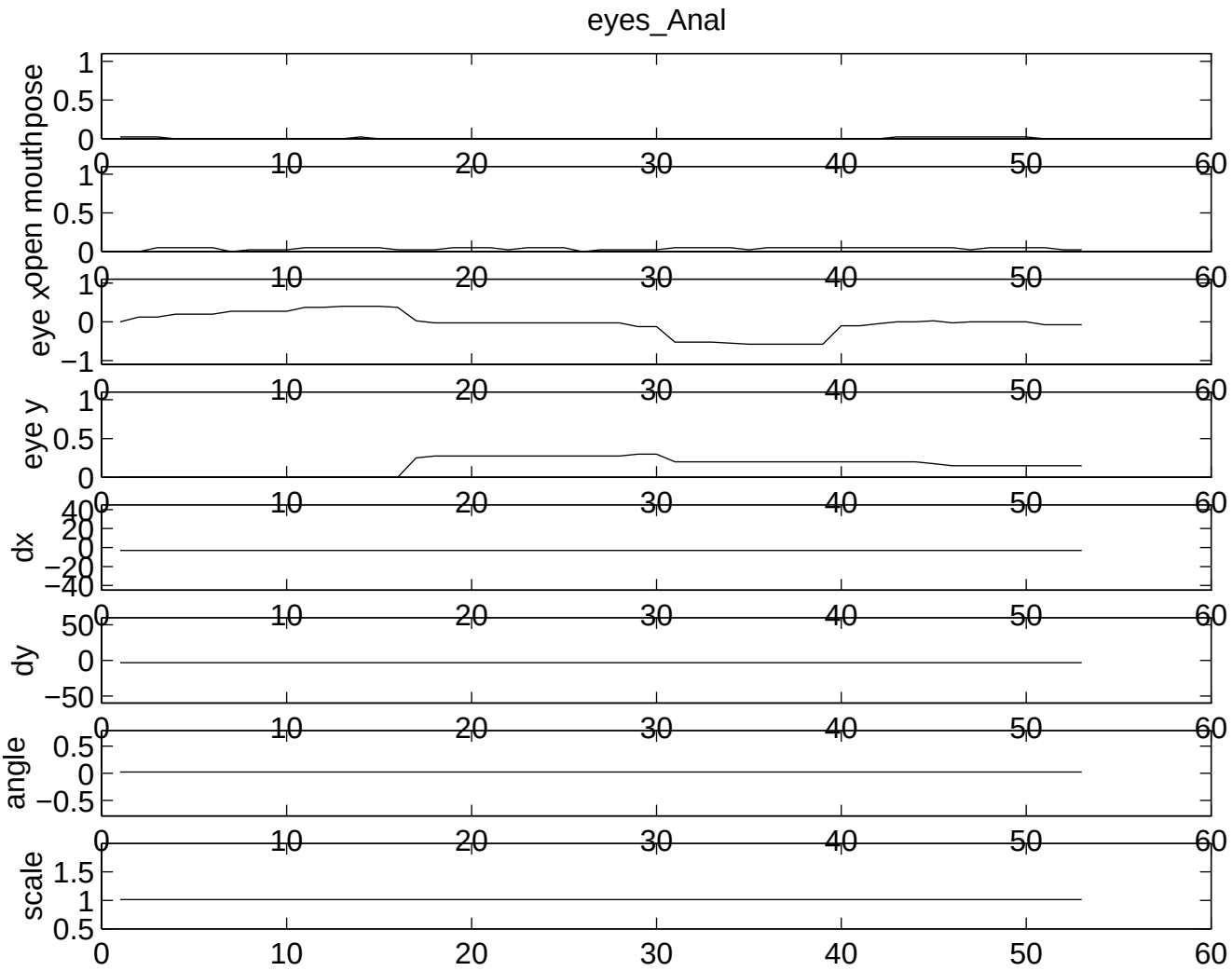


Figure 3-20: The parameters extracted from the previous novel sequence.

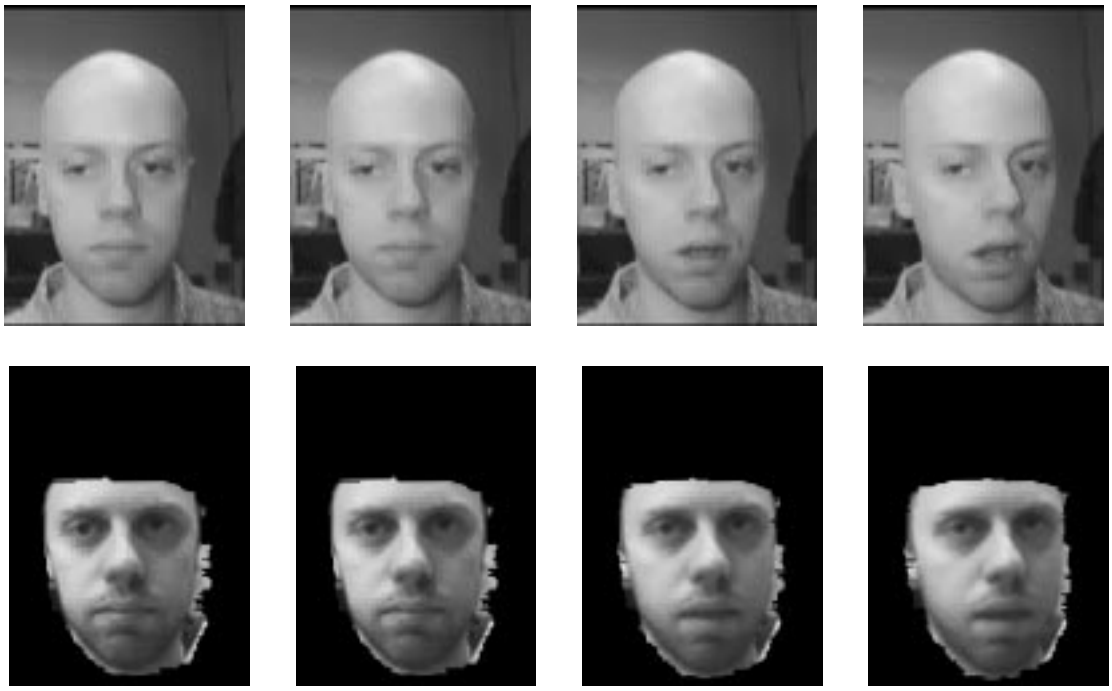


Figure 3-21: A novel sequence with combined pose, mouth, and eye movement, juxtaposed along with the synthesized sequence.

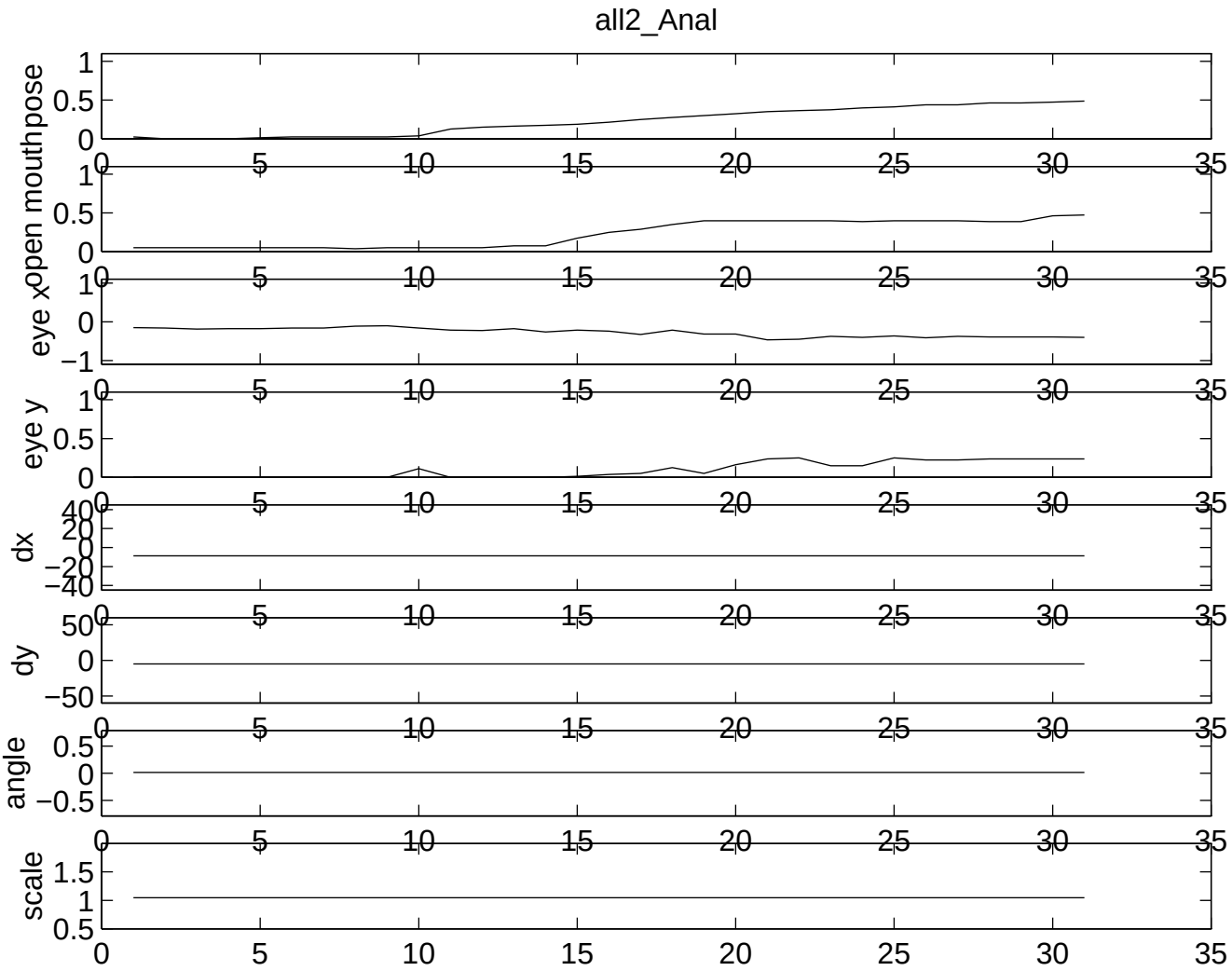


Figure 3-22: The parameters extracted from the previous novel sequence.

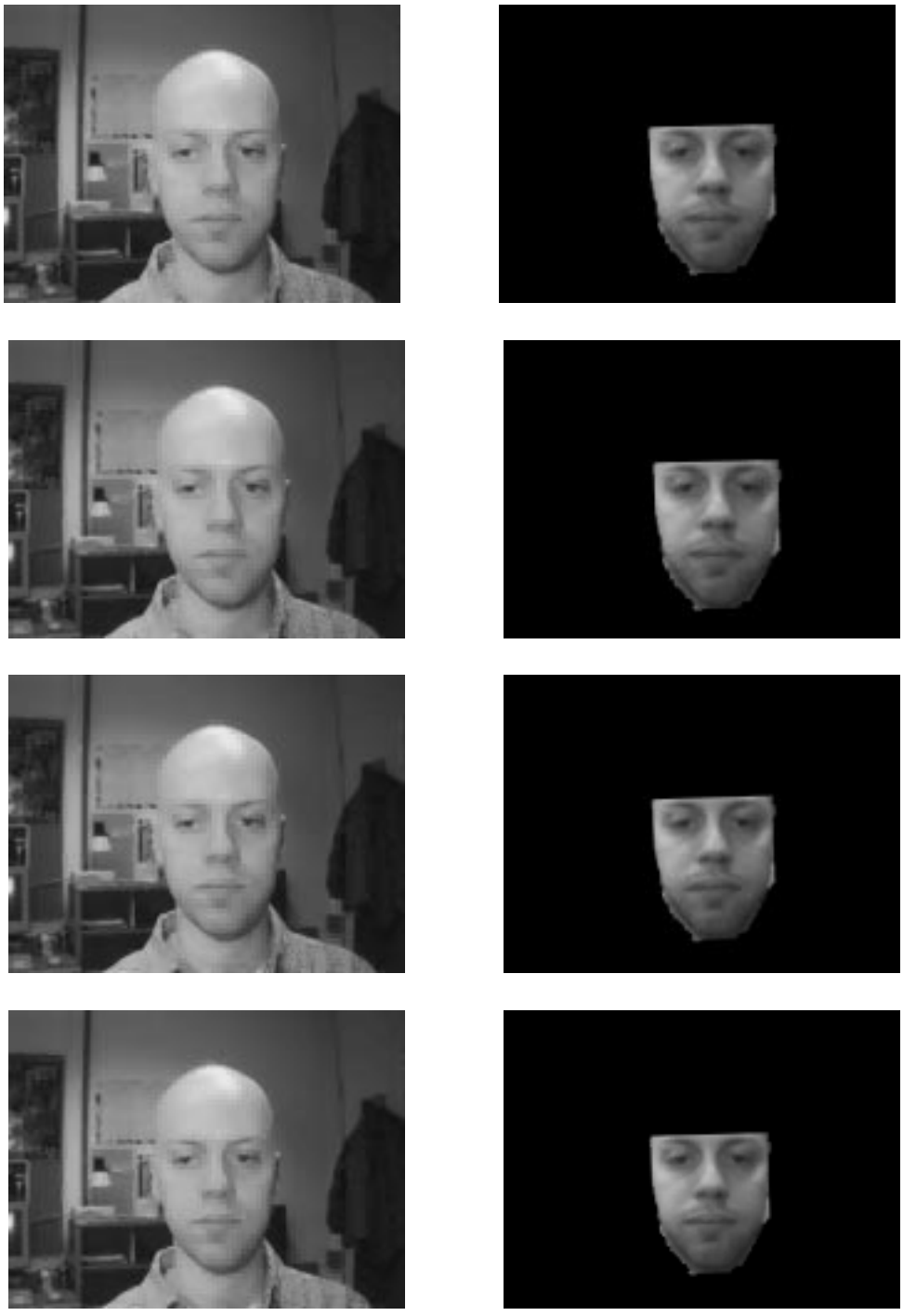


Figure 3-23: A novel sequence with affine movement in which the scale of the head decreases, juxtaposed along with the synthesized sequence.

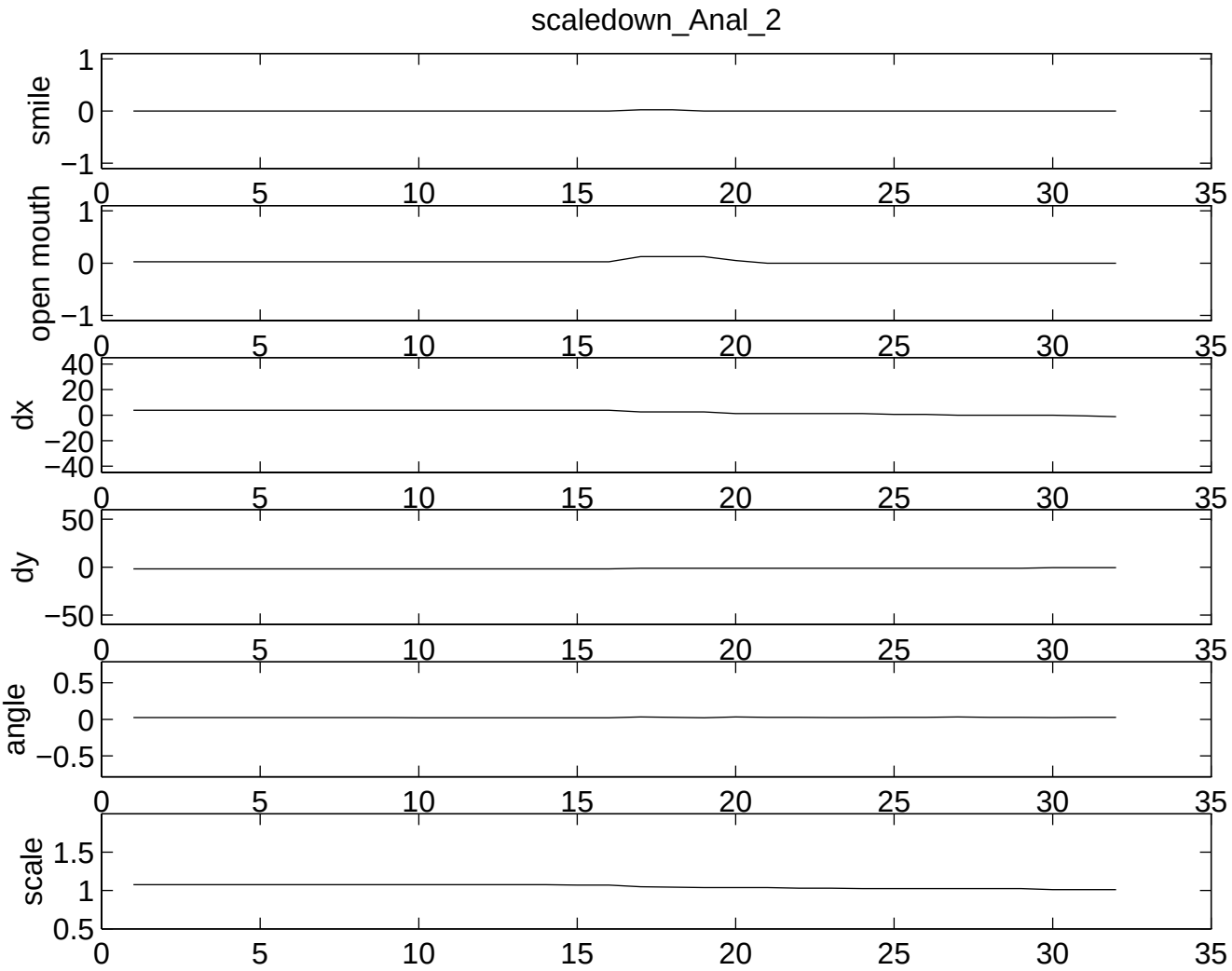


Figure 3-24: The parameters extracted from the previous novel sequence.

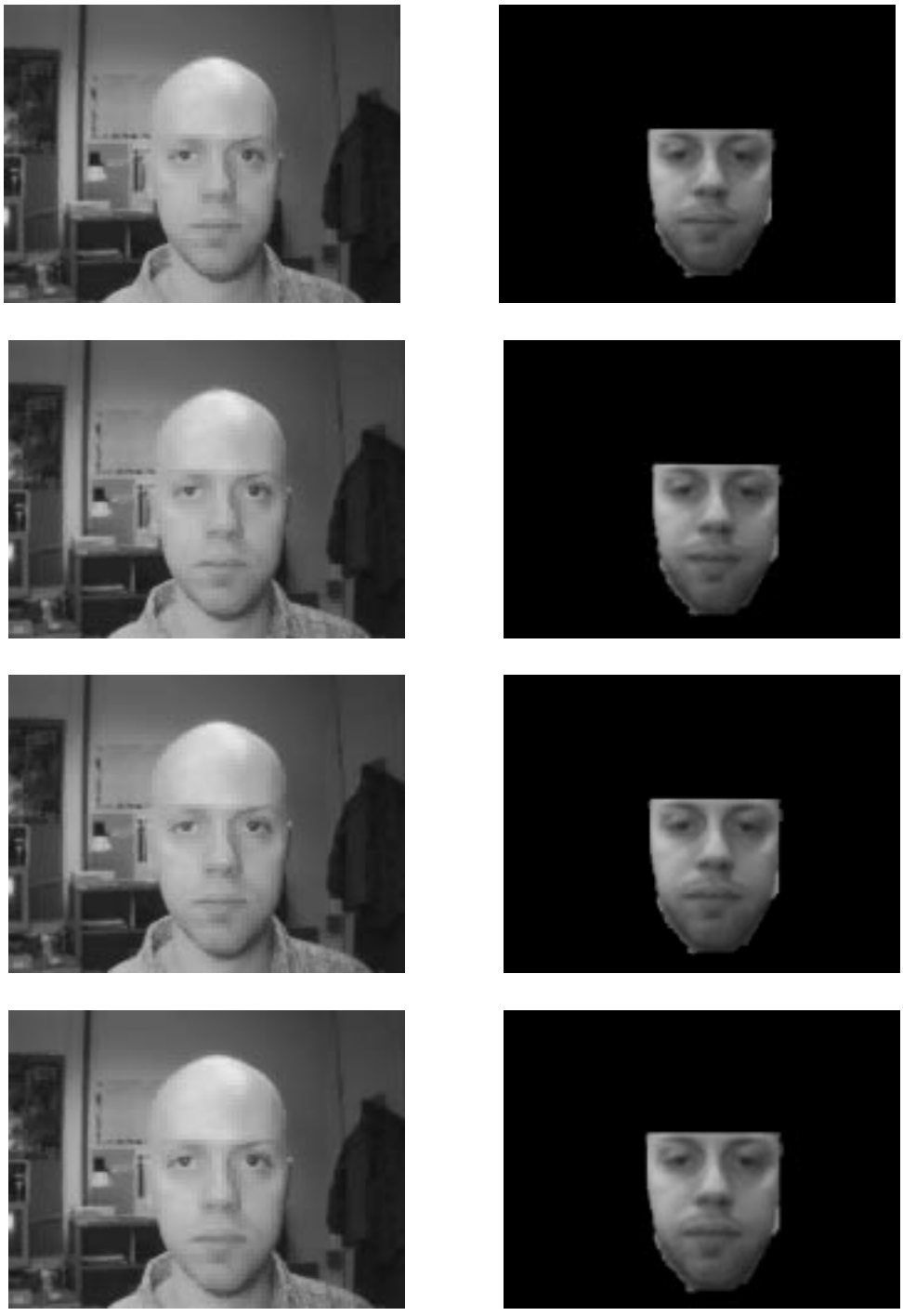


Figure 3-25: A novel sequence with affine movement in which the scale of the head increases, juxtaposed along with the synthesized sequence.

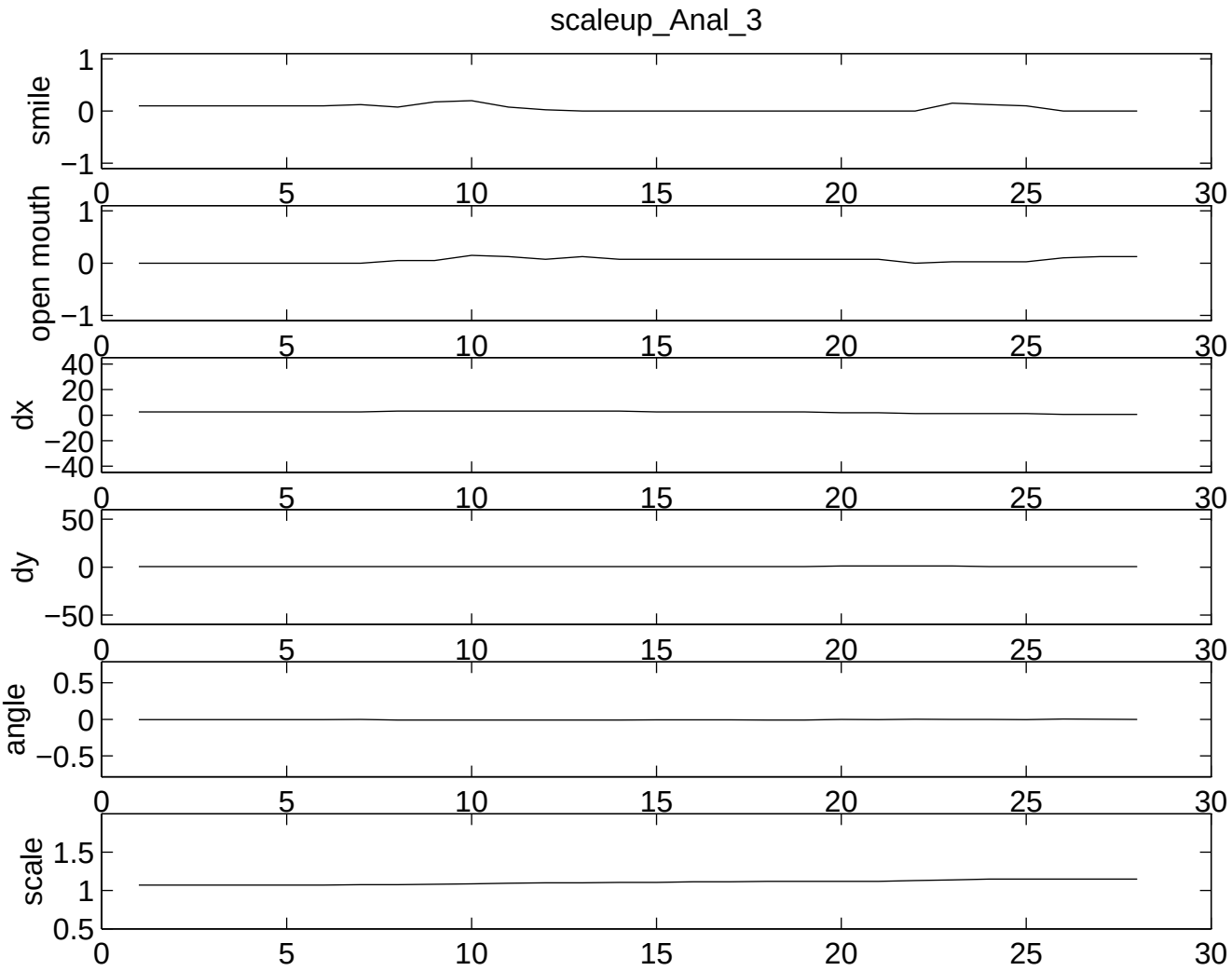


Figure 3-26: The parameters extracted from the previous novel sequence.

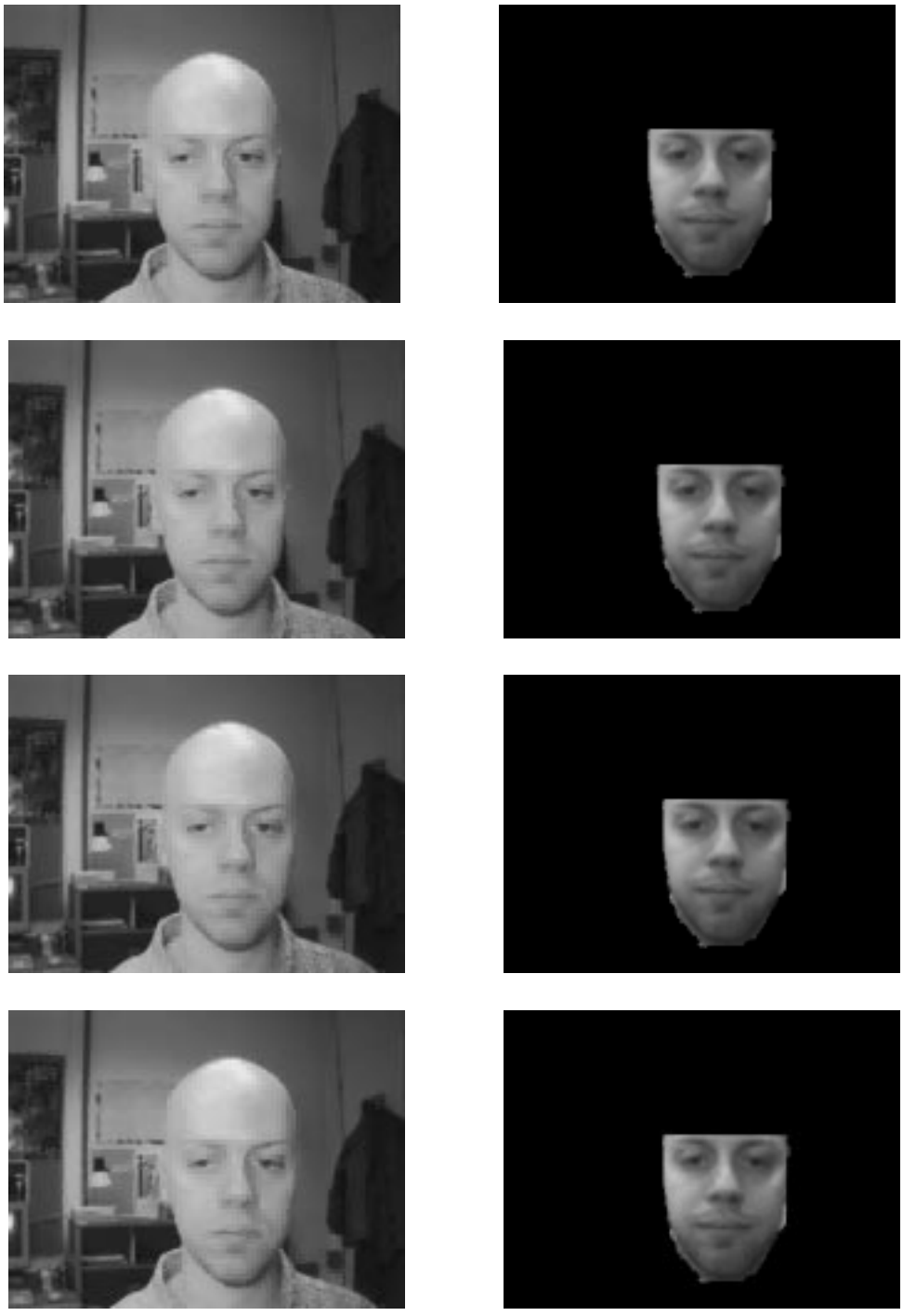


Figure 3-27: A novel sequence with affine movement in which the head translates, juxtaposed along with the synthesized sequence.

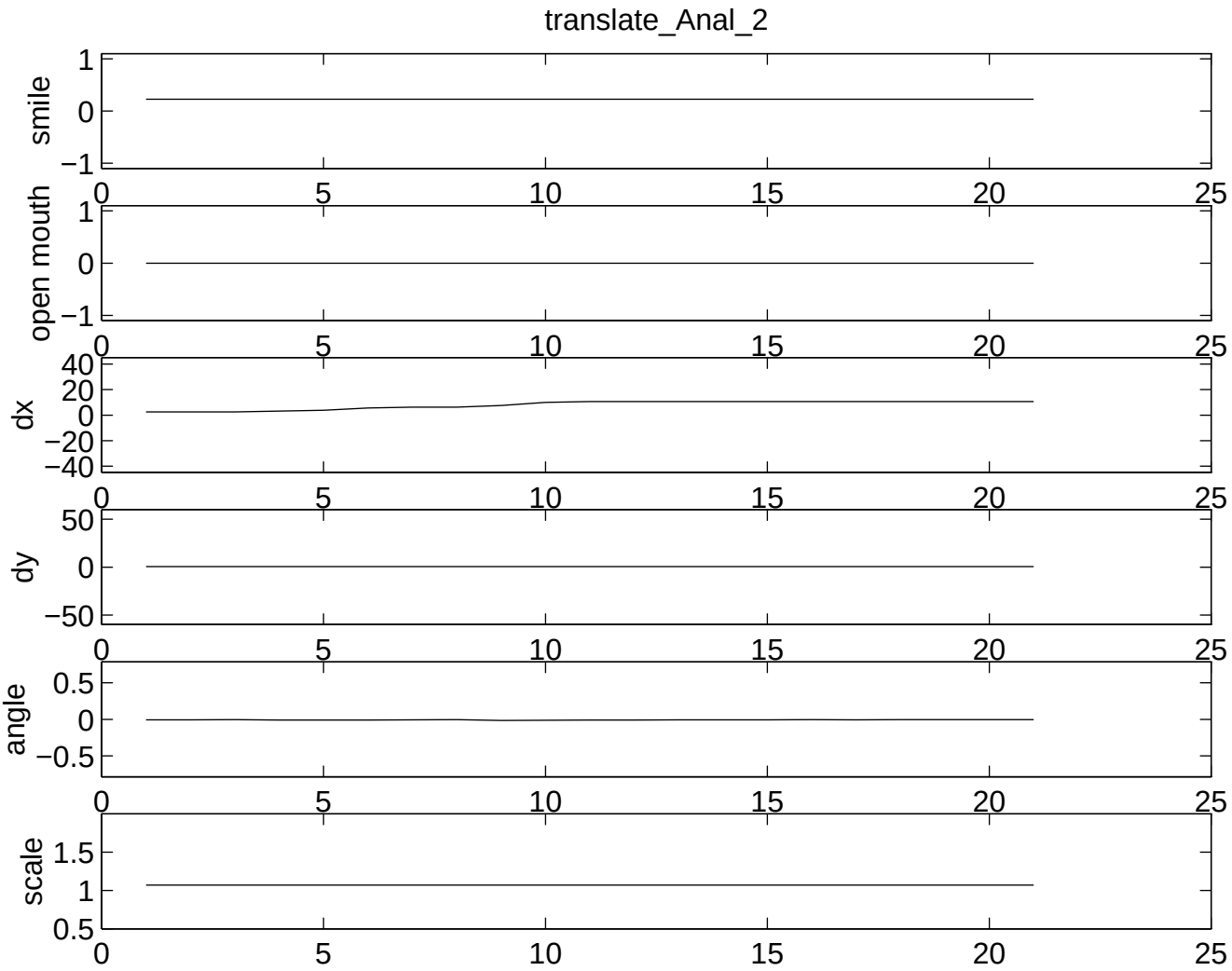


Figure 3-28: The parameters extracted from the previous novel sequence.

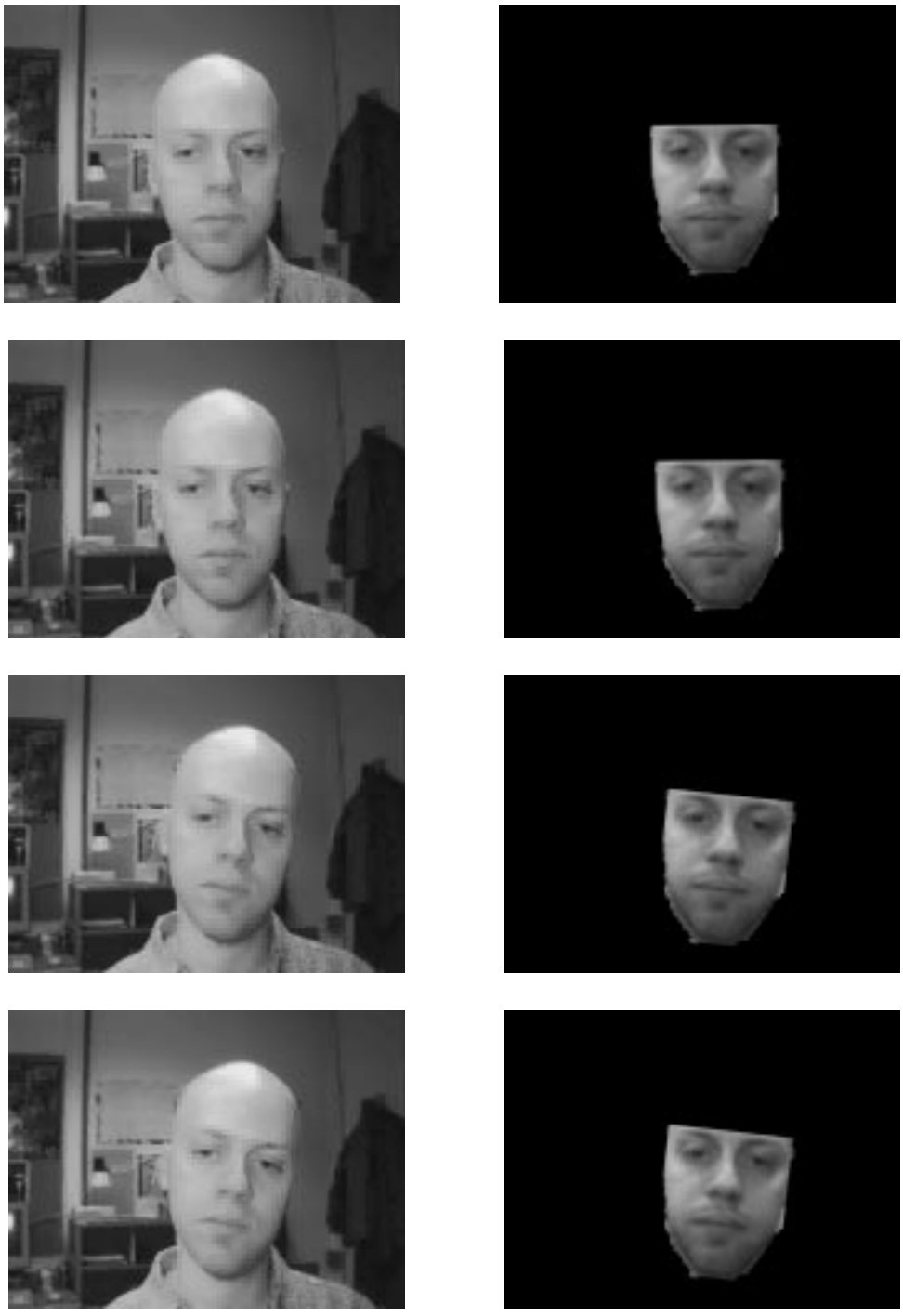


Figure 3-29: A novel sequence with affine movement in which the head rotates, juxtaposed along with the synthesized sequence.

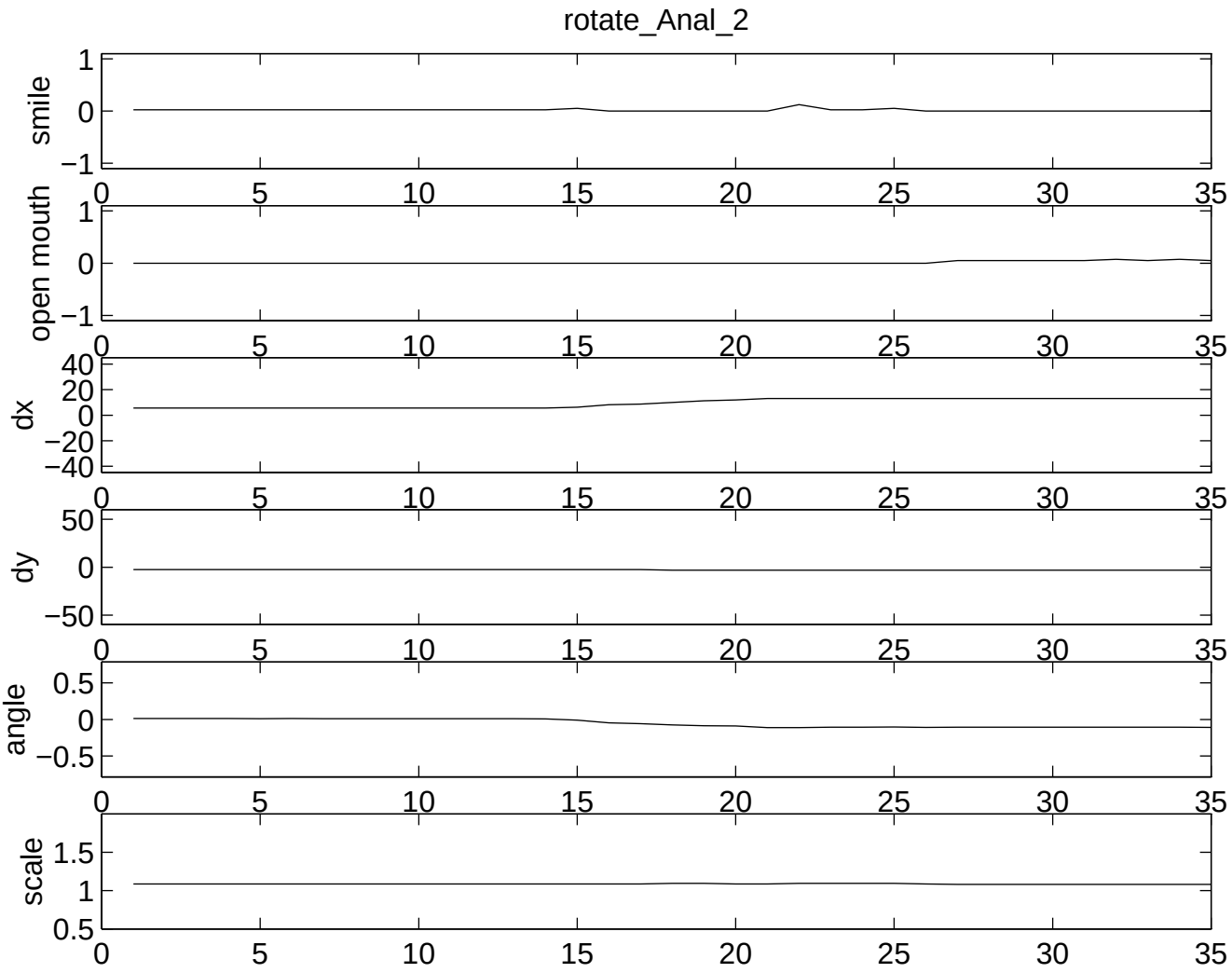


Figure 3-30: The parameters extracted from the previous novel sequence.

Bibliography

[1] 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.

[2] K. Aizawa and Thomas Huang. Model-based image coding: Advanced video coding techniques for very low bit-rate applications. *Proceedings of the IEEE*, 83:259–271, 1995.

[3] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92 Proceedings*, pages 35–42, Chicago, IL, 1992.

[4] J.R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, Princeton, New Jersey, April 1990.

[5] 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.

[6] 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.

[7] Shenchang Eric Chen. Quicktime vr - an image-based approach to virtual environment navigation. In *SIGGRAPH '95 Proceedings*, pages 29–37, Los Angeles, CA, August 1995.

[8] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93 Proceedings*, pages 279–288, Anaheim, CA, August 1993.

[9] P. Ekman and W.V. Friesen. *Facial Action Coding System*. Consulting Psychologists Press, Inc., Palo Alto, CA, 1977.

[10] 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.

[11] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[12] H.G.Musmann, M.Hoetter, and J. Ostermann. Object-oriented analysis-synthesis coding of moving images. *Signal Processing : Image Communication*, 1:117–138, 1991.

[13] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[14] 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.

[15] James Kuch and Thomas Huang. Vision based hand modelling and tracking for virtual teleconferencing and telecollaboration. In *Proceedings of the International Conference on Computer Vision*, pages 666–671, Boston, Massachusetts, June 1995.

[16] Rakesh Kumar, P. Anandan, Michal Irani, James Bergen, and Keith Hanna. Representation of scenes from collections of images. In *IEEE Workshop on Representation of Visual Scenes*, Cambridge, MA, June 1995.

[17] S. Laveau and O. Faugeras. 3-d scene representation as a collection of images and fundamental matrices. Technical Report Technical Report No. 2205, INRIA, 1994.

[18] Andrew Lippman. Movie-maps: An application of the optical videodisc to computer grpahics. In *Siggraph '80 Proceedings*, pages 32–41, Los Angeles, CA, 1980.

[19] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH '95 Proceedings*, Los Angeles, CA, 1995.

[20] F. I. Parke. A model for human faces that allows speech synchronised animation. *Computers and Graphics*, 1:1–4, 1975.

[21] Donald Pearson. Developments in model-based coding. *Proceedings of the IEEE*, 83:892–906, 1995.

[22] Tomaso Poggio and Roberto Brunelli. A novel approach to graphics. A.I. Memo No. 1354, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[23] 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.

[24] 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.

[25] Shimon Ullman and Ronen Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[26] John Wang and Edward Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing, Special Issue:Image Sequence Compression*, 3(5):625–638, September 1994.

[27] W. J. Welsh. Model-based coding of moving images at very low bitrates. In *Proceedings of the International Picture Coding Symposium*, page paper 3.9, Stockholm, Sweden, 1987.

[28] Tomáš Werner, Roger David Hersch, and Václav Hlaváč. Rendering real-world objects using view interpolation. In *Proceedings of the International Conference on Computer Vision*, Cambridge, MA, June 1995.

[29] J. F. S. Yau and N. D. Duffy. A texture-mapping approach to 3-d facial image synthesis. In *Proceedings of the 6th Annual Eurographics Conference*, Sussex, UK, 1988.