Using Recorded Motion for Facial Animation

by

Tony Kobayashi

B.Sc., Carnegie Mellon University, 1994

AN ESSAY SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

I accept this essay as conforming
to the required standard

The University of British Columbia

April 1997

Abstract

In this essay we discuss two implementations of facial animation. Both were done using Dr. David Forsey's hierarchical B-Spline editor, Dragon[5]. One animation was driven by motion captured data. The second was driven by a prototype for the MPEG-4 encoding of facial animation.

Contents

Acknowledgements

I'd like to thank my advisor, Dr. David Forsey for providing the Dragon editor, his expertise, and his deadlines.

Tony Kobayashi

The University of British Columbia

April 1997

To Tracy, my love.

Chapter 1

Introduction

In this essay we discuss two implementations of facial animation in Dr. David Forsey's hierarchical B-Spline editor, Dragon [5]. The first implementation was driven by motion captured data and the second implementation was driven by test data for the upcoming MPEG-4 facial animation encoding standard.

The problem of animating a face can be broken down into two stages – modelling and animation. Dr. David Forsey's hierarchical B-spline modeller, Dragon, was used in this project because it has several characteristics that are useful in both areas.

1.1 Modelling

There are a number of ways to model a face. A set of polygons would be the simplest, but real faces do not consist of flat surfaces and sharp angles. Since they are curved, curved surfaces like B-spline surfaces would make better candidates.

B-spline surfaces are defined by a grid of control vertices. Each set of 4×4 vertices defines a single patch. The grid does not have to be uniformly spaced, but

each row and column must have the same number of points. This poses a problem when modelling a face because heads have areas of fine detail – the eyes, nose, and mouth – and coarse detail – the forehead, cheeks, and back. To model a face with high detail around the eyes in a traditional B-spline modeller would require very small grid spacing along the column of the eye which is essentially wasted.

The Dragon editor overcomes this problem by allowing patches to be defined locally. The grid is no longer a flat global structure. Instead, it is replaced by a hierarchy which allows finer details to be added to an existing surface. Figure 1.1 shows a simple surface at level 0, level 1, and level 2. The left image is of the simplest B-spline patch which is defined by 16 control vertices at level 0. In the middle image, the patch has been divided into 4 patches which are each defined by 16 control vertices. But many of them overlap and so there are only 25 distinct control vertices at level 1. The control vertex in the middle was moved. In the right image, the two patches in the back have been subdivided creating 8 distinct patches at level 2. Once again, many of the control vertices overlap and so there are only 35 distinct control vertices at level 2. The two vertices at the back corners have been moved.

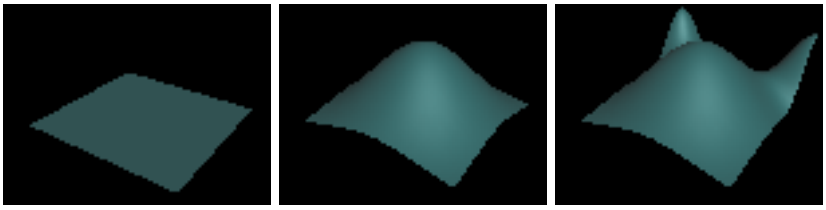

Figure 1.1: Multiple resolutions of a simple surface in the Dragon editor

Instead of storing vertices, Dragon only stores non-zero offsets. Thus, the surface in figure 1.1 only needs to store 16 (offsets at level 0) + 1 (offset at level 1) + 2 (offsets at level 2) = 19 offsets instead of 16 + 25 + 35 = 76 control vertices. Since

offsets are stored instead of absolute coordinates, changes made at level 0 would be reflected in the level 1 and level 2 control vertices as well. Similarly, a face would have multiple resolutions and changing any of the control vertices at level 0 would affect the lower resolution control vertices.

1.2 Animation

Animating a face is difficult because even simple movements may correspond to many control vertices moving in complex ways. Consider a bicycle with its front wheel spinning. The entire motion can be easily described in terms of the speed of rotation and its axis. But how would one describe the motion of a person's lips moving when they open their mouth? One would have to set up a complicated description of how far control vertices on the lips move as a function of the mouth opening angle or specify areas of influence for each control vertex.

The Dragon editor intrinsically specifies these areas of influence. Each patch is defined by 16 control vertices and so each control vertex (excluding ones near the boundary of the surface) affects 16 patches. But unlike traditional B-spline modellers where control vertices are specified as a flat grid structure, Dragon defines a hierarchy of control vertices. The result is that we can specify different size areas of influence by choosing control vertices at different levels of the hierarchy.

Consider the problem of animating the opening of a jaw [4]. Figure 1.2 shows a face model at level 0 and level 4. To open the jaw in a traditional modeller would involve modifying over a hundred control vertices. But in Dragon, we can modify just twelve control vertices at level 0. Since Dragon works with offsets instead of absolute coordinates for control vertices, all lower level control vertices will automatically be modified appropriately. Figure 1.3 shows the open mouth version with twelve offsets

changed at level 0 and the resulting surface at level 4.

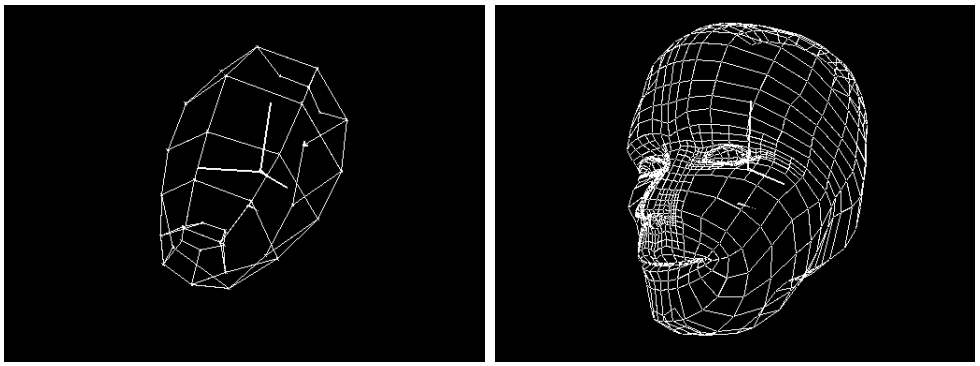

Figure 1.2: A head model in the Dragon editor

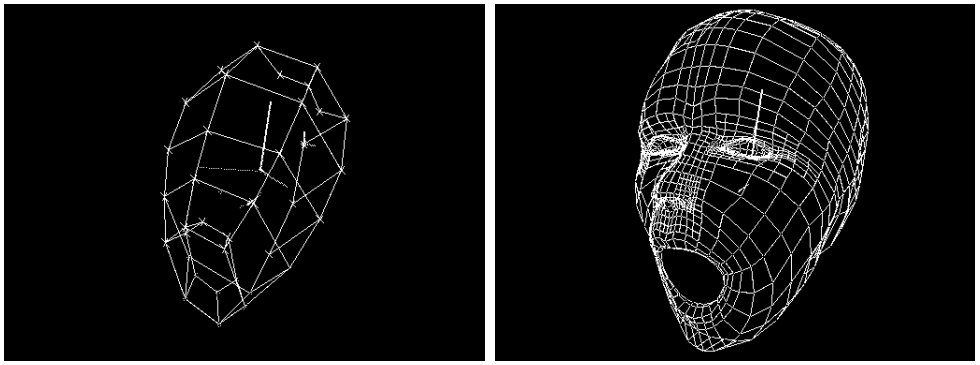

Figure 1.3: Head model with mouth opened

The twelve control vertices at level 0 can be attached to a segment in Dragon. An additional advantage to using Dragon is that animation "actions" can be created and that they can be made relative. In this way, one action might represent the opening and closing of the jaw and another might represent the shifting of the jaw. Both actions affect the same twelve vertices, but since they are defined to be relative, they can be applied independently. Also, actions at lower levels (such as a pursing of the lips done through offsets at level 3) would be completely independent as well.

Chapter 2

Related Works

There are many approaches to animating faces [23]. There are physical based models which try to model the underlying muscle structure [12] [16] [11] [17] [27] [26] [30] [28]. There are image based approaches [34] [31]. Some papers focus on expressions [10] [2] [22] [24] and some focus on lip-synching [14] [15] [9] [8] [6] [7] [18] [19] [29] [32]. There are also approaches centered on the models [21] [20] [25] [33].

Chapter 3

Head Models

To create a facial animation, we need two pieces of information: a model of a head and a specification for the animation. Three head models were provided for use with this project, a human head, a dog's head, and a dragon's head.

3.1 Human Head

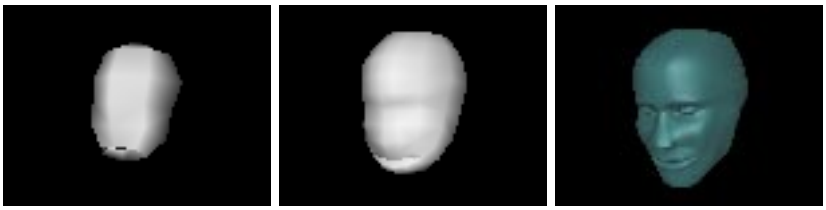

Figure 3.1: The human head model in the Dragon editor

The human head was converted to a Dragon model from an existing model in Alias format. Figure 3.1 shows the human head model at different resolutions. The model contains 1083 offsets among six levels of detail.

3.2 Dog Head

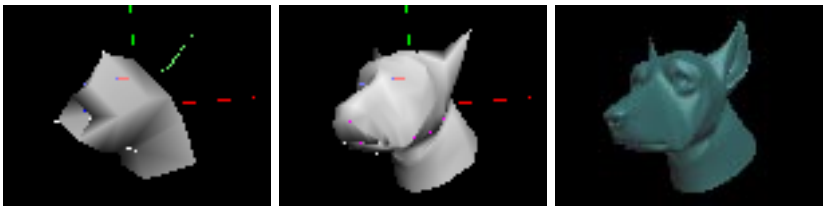

Figure 3.2: The dog head model in the Dragon editor

The dog head was created by Goesta Struve-Dencher, a professional computer artist. The entire model was created within the Dragon editor in a few hours. Figure 3.2 shows the dog head model at different resolutions. The model contains 736 offsets among seven levels of detail.

3.3 Dragon Head

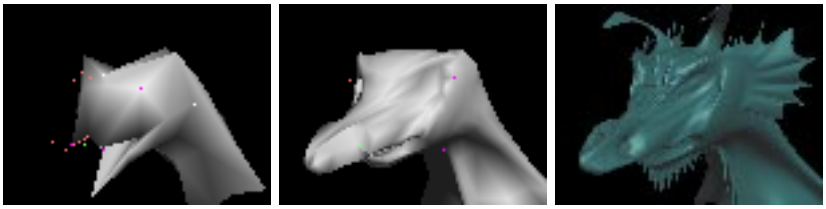

Figure 3.3: The dragon head model in the Dragon editor

The dragon head was created by Dr. Forsey as part of his PhD dissertation work with hierarchical B-Spline surfaces. Figure 3.3 shows the dragon head model at different resolutions. The model contains 506 control vertices among nine levels of detail.

Chapter 4

Animation Data

The second piece of information required is the animation itself. Two types of data were provided. One was recorded information of a real head moving and the other was generated for use with the upcoming MPEG-4 standard.

4.1 Motion Captured Data

The motion captured data was gathered by placing 49 markers on a human subject's face and recording each marker's position in three dimensional space at a rate of 100 frames per second. The data was then filtered to eliminate noise, decimated to 24 frames per second, and finally separated into three files: a neutral pose file which specified the marker positions of the face at rest, a vector file which specified the location of each of the 49 markers with respect to its neutral pose at each frame, and a global orientation file which specified the orientation of the head at each frame.

Theoretically, this should produce the most realistic data. But practically, motion captured data tends to be very noisy. This results in very jerky animations if the data is used directly to drive the animation. Also, the given files did not seem

to be accurately separated between the vector file and the global orientation file. Global orientation was still quite visible even when the vector file was viewed by itself.

4.2 MPEG-4.0 Data

The upcoming MPEG-4 [3] standard includes an encoding for specifying facial animation. Currently, it defines 68 facial action parameters (FAPs) divided into ten groups. Table 4.1 describes the ten different groups and tables 4.3 and 4.4 list the 68 facial action parameters currently defined. Table 4.2 describes the units used for the FAPs. Most of the FAPs are numerical values between -1000 and 1000 which specify how far a given facial feature moves in a given direction. Each FAP has a unit, direction, and quantization step size associated with it.

Consider FAP number 4 – "lower_t_midlip". This represents the downward motion of the middle of the top lip. Its unit is "MNS" and the direction is "down" which indicates that a value of 1000 for FAP 4 corresponds to the middle of the top lip being one mouth-to-nose unit down from its neutral position. That is, the distance from the lip's neutral position to its current position is the same as the distance from the mouth to the nose. A value of -1000 for FAP 4 would correspond to the lip moving up one mouth-to-nose unit. In other words, the upper lip would be at the bottom of the nose.

The MPEG-4 facial animation standard was initially designed to provide efficient transfer of facial animations for applications like teleconferencing. So it is somewhat biased towards a frontal view of the face. As a result, many of the features which have lateral and vertical actions associated with them do not have depth actions. However, the MPEG-4 data has one advantage over the motion captured

data. The MPEG-4 data is already specified proportional to various distances on the face. The distances used are given in table 4.2.

Group Number	Group Description
1	visemes and expressions
2	jaw, innerlip, chin
3	eyeballs, pupils, eyelids
4	eyebrow
5	cheeks
6	tongue
7	head rotation
8	outer lip positions
9	nose
10	ears

Table 4.1: Facial Action Parameter Groups

FAP Unit	Description
deg	degrees
ENS	vertical distance between eyes and nose
ES	distance between pupils of eyes
MNS	distance from mouth to nose
MW	width of mouth

Table 4.2: Facial Action Parameter Units

FAP num	FAP name	FAP units	Uni- / Bi-dir	FAP motion	Group	Quant step size
1	viseme	na	na	na	1	na
2	expression	na	na	na	1	na
3	open_jaw	MNS	U	down	2	8
4	lower_t_midlip	MNS	B	down	2	5
5	raise_b_midlip	MNS	B	up	2	5
6	stretch_l_cornerlip	MNS	B	left	2	5
7	stretch_r_cornerlip	MNS	B	right	2	5
8	raise_t_lip_lm	MNS	B	up	2	5
9	raise_t_lip_rm	MNS	B	up	2	5
10	lower_b_lip_lm	MNS	B	down	2	5
11	lower_b_lip_rm	MNS	B	down	2	5
12	raise_l_cornerlip	MNS	B	up	2	5
13	raise_r_cornerlip	MNS	B	up	2	5
14	thrust_jaw	MNS	U	forward	2	3
15	shift_jaw	MNS	B	right	2	3
16	push_b_lip	MNS	B	forward	2	3
17	push_t_lip	MNS	B	forward	2	3
18	depress_chin	MNS	U	up	2	1
19	close_t_l_eyelid	ENS	B	down	3	3
20	close_t_r_eyelid	ENS	B	down	3	3
21	close_b_l_eyelid	ENS	B	up	3	2
22	close_b_r_eyelid	ENS	B	up	3	2
23	yaw_l_eyeball	deg	B	right	3	1
24	yaw_r_eyeball	deg	B	right	3	1
25	pitch_l_eyeball	deg	B	down	3	1
26	pitch_r_eyeball	deg	B	down	3	1
27	thrust_l_eyeball	deg	B	forward	3	1
28	thrust_r_eyeball	deg	B	forward	3	1
29	dilate_l_pupil	ES	U	na	3	1
30	dilate_r_pupil	ES	U	na	3	1
31	raise_l_i_eyebrow	ENS	B	up	4	4
32	raise_r_i_eyebrow	ENS	B	up	4	4
33	raise_l_m_eyebrow	ENS	B	up	4	4
34	raise_r_m_eyebrow	ENS	B	up	4	4
35	raise_l_o_eyebrow	ENS	B	up	4	4
36	raise_r_o_eyebrow	ENS	B	up	4	4
37	squeeze_l_eyebrow	ES	B	right	4	2
38	squeeze_r_eyebrow	ES	B	left	4	2

Table 4.3: MPEG-4 Facial Action Parameters, Groups 1–4

FAP num	FAP name	FAP units	Uni- / Bi-dir	FAP motion	Group	Quant step size
39	puff_l_cheek	ES	B	left	5	4
40	puff_r_cheek	ES	B	right	5	4
41	lift_l_cheek	ENS	U	up	5	4
42	lift_r_cheek	ENS	U	up	5	4
43	shift_tongue_tip	MW	B	right	6	2
44	raise_tongue_tip	MW	B	up	6	2
45	thrust_tongue_tip	MW	B	forward	6	2
46	raise_tongue	MW	B	up	6	2
47	tongue_roll	deg	U	up	6	2
48	head_pitch	deg	B	up	7	1
49	head_roll	deg	B	right	7	1
50	head_yaw	deg	B	right	7	1
51	lower_t_midlip _o	MNS	B	down	8	5
52	raise_b_midlip_o	MNS	B	up	8	5
53	stretch_l_cornerlip_o	MNS	B	left	8	5
54	stretch_r_cornerlip_o	MNS	B	right	8	5
55	raise_t_lip_lm _o	MNS	B	up	8	5
56	raise_t_lip_rm _o	MNS	B	up	8	5
57	lower_b_lip_lm_o	MNS	B	down	8	5
58	lower_b_lip_rm_o	MNS	B	down	8	5
59	raise_l_cornerlip_o	MNS	B	up	8	5
60	raise_r_cornerlip _o	MNS	B	up	8	5
61	stretch_l_nose	ENS	B	left	9	3
62	stretch_r_nose	ENS	B	right	9	3
63	raise_nose	ENS	B	up	9	3
64	bend_nose	ENS	B	right	9	3
65	raise_l_ear	ENS	B	up	10	2
66	raise_r_ear	ENS	B	up	10	2
67	pull_l_ear	ENS	B	up	10	2
68	pull_r_ear	ENS	B	up	10	2

Table 4.4: MPEG-4 Facial Action Parameters, Groups 5–10

Chapter 5

Techniques Used

Different techniques were used for the two types of animation data given. For the motion captured data, the heads were animated by moving control vertices directly as specified in the vector file. For the MPEG-4 data, the heads were animated by moving control vertices as specified by each FAP value and its definition.

In both cases, care had to be taken in not only which control vertices were used, but also in what level in the hierarchy they were chosen from. In general, the highest level possible would give the smoothest and most natural looking animation.

5.1 Motion Captured Data

For this part of the project, a set of 49 segments were defined for each of the Dragon heads which would correspond to each of the 49 markers used in the motion captured data. Next, each of these segments was attached to several nearby control vertices at appropriate levels. Finally a program was written in C++ to convert the motion captured data vector file into an animation file format that Dragon could understand. This animation file would move the segments which would move the

control vertices which would move the surfaces and the result is a facial animation.

This procedure worked well with the human head which had roughly the same shape and dimensions of the human subject used to gather the motion capture data. But problems arose when trying to apply this procedure to a vastly different shaped head – for example, the dog head.

When trying to animate a dog's head with human head animation, unnatural movements would be created because the same coordinate system was used in both animations. Consider a human smile – the lips are pulled outward and slightly upward. Figure 5.1 shows what those two actions would look like if they were mapped directly to a dog's head. This looks very unnatural. A better interpretation would be to pull the lips back and slightly upward. Figure 5.2 shows a more reasonable looking smiling dog.

Figure 5.1: Direct mapping of human action to a dog's head

In order to generate appropriate animations, the animations must take into consideration the shape of each head. This can either be done manually or automatically. One idea for compensating for the different geometries automatically is to extend a 2D morphing algorithm[1] into 3D[13]. Using this method, an animation for a human head could be "morphed" into an animation for for a dog's head.

Figure 5.2: Desired mapping of human action to a dog's head

5.2 MPEG-4.0 Data

In contrast to raw motion captured data, the MPEG-4 facial animation parameters are scalar values. An action is created for each parameter manually. Different facial shapes are compensated for by adjusting the action created. For example, a parameter may originally indicate a horizontal movement but the action created may be a combination of a horizontal movement and a change in depth.

Theoretically, the same process could have been applied to the motion captured data by simply splitting up each of the vector values into the corresponding coordinates. But the motion captured data uses 49 markers and the MPEG-4 data uses the equivalent of about 25 markers. Also, some of the MPEG-4 "markers" use only one or two of the 3 coordinates. For example, there are "stretch_l_cornerlip" and "raise_l_cornerlip" FAPS (numbers 6 and 12) which correspond to horizontal and vertical movement. But there is no corresponding depth movement FAP for the left corner lip. The result is that one would have to define nearly two and a half times as many actions to create an analogous setup with the motion captured data. MPEG-4 sacrifices some accuracy in favor of bandwidth — only the most commonly

changing coordinates are used.

Also, the MPEG-4 facial animation parameters are given relative to certain facial features. For example, the horizontal movements of the lips are given in units of the mouth width at a neutral pose. Once again, the same thing could be accomplished with the motion captured data by measuring the neutral pose face and running the vector data through a preprocessor. But with the MPEG-4 data, there is one less thing for the animator to worry about.

Once the 68 FAP actions are created in Dragon for a given model, the animation process is quite straightforward. Each of the original MPEG-4 facial animation parameters drives one action and each action in turn affects a number of control vertices which animates the surface. The same global animation can be reused on any other Dragon model which has these 68 FAP actions defined. There is no need for translation because each individual FAP actions was customized to the model it was created for.

Chapter 6

Results

In general, the images look better for the MPEG-4 implementation than the motion captured data implementation. But this has more to do with the quality of the data than any inherent differences.

6.1 Motion Captured Data

Figure 6.1 shows the positions of the 49 markers used in the motion captured data. Edges were arbitrarily drawn between markers and the resulting triangle faces were made visible to see the structure more clearly. Note that the markers are fairly evenly spaced. The left image is the neutral position of the 49 markers, the middle image is frame 82 of 190 from the test data, and the right image is frame 110 of 190.

Figure 6.2 shows the same 49 markers as segments on the dog head model in neutral position. These segments were each attached to several control vertices at differing levels. Then, the segments were animated directly by the vector data given which would move the attached control vertices. Different results would be obtained depending on what levels the attachments were made at and how many of

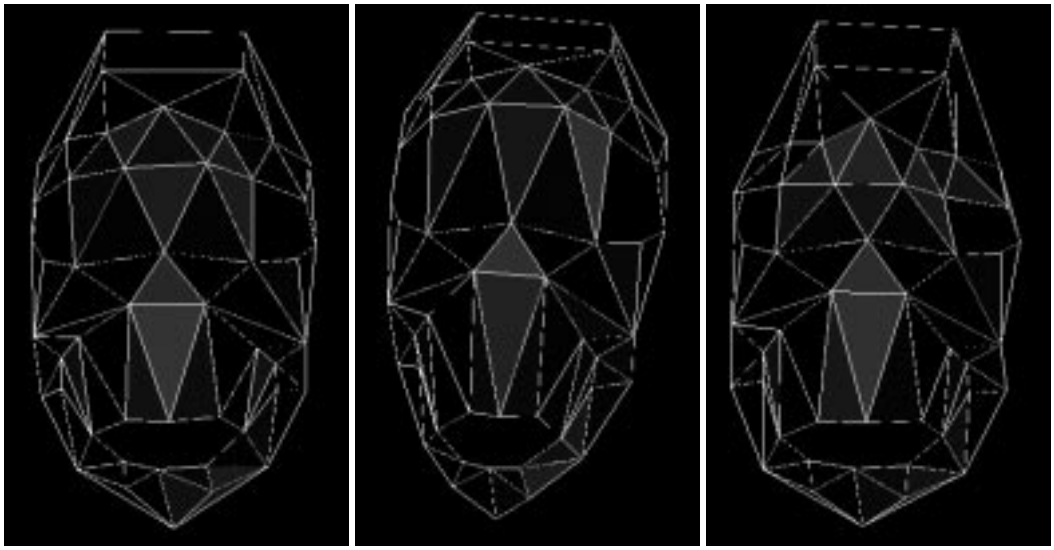

Figure 6.1: Motion Captured data — neutral, frame 82, and frame 110

the 49 segments were actually used.

Figure 6.3 shows the neutral position, frame 82, and frame 110 which result if the attachments are made at low levels (more detailed). Figure 6.4 shows the same frames if the attachments are made at higher levels (more coarse). Also, only one fourth of the segments were actually used in figure 6.4. In both sets of the images, the faces seem unnaturally deformed because rotation was not properly filtered out of the data. The first set is done at too low a level which is especially evident in frame 82 around the mouth. The result is that it looks like the lip is being pulled down instead of the whole jaw. In figure 6.4, the mouth looks much more natural.

Note that even though we used one fourth as many segments in figure 6.4, the images actually look better. Fewer well chosen markers can be better than too many markers, especially when the data is very noisy.

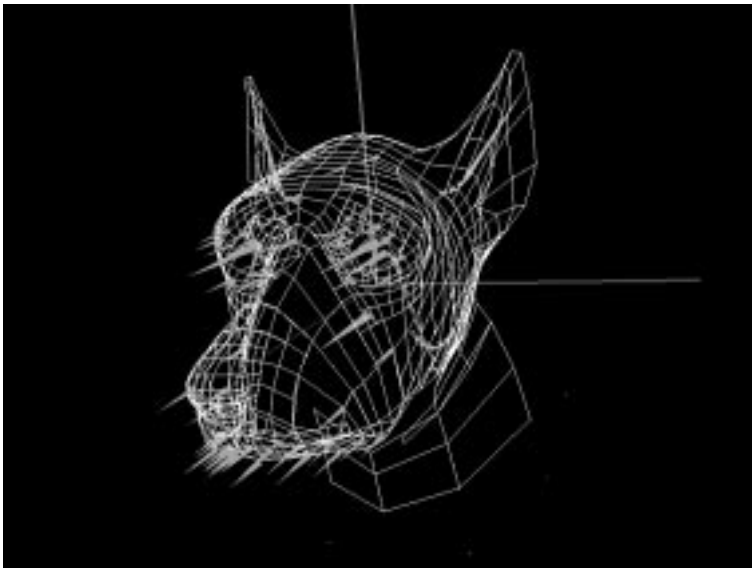

Figure 6.2: Markers on the dog's head

6.2 MPEG-4.0 Data

The MPEG-4.0 Data given only used facial animation parameter (FAP) groups 2 and 7 — the jaw, innerlip, chin group and the head rotation group. But of these two groups, only group 2 had non-zero data.

Each of the 16 FAPs in group 2 (FAP 3 through FAP 18) was then encoded into an action in the Dragon animation system. The difference between a segment associated with a group of control vertices and an action associated with a group of control vertices is that a segment moves in 3 dimensional space and an action moves in 1 dimensional time. Three actions would need to be created (one for the X coordinate, one for the Y coordinate, and one for the Z coordinate) to simulate the analogous segment.

The main advantage of using an action instead of segments to drive the animation is more control. The control vertices associated with an action do not

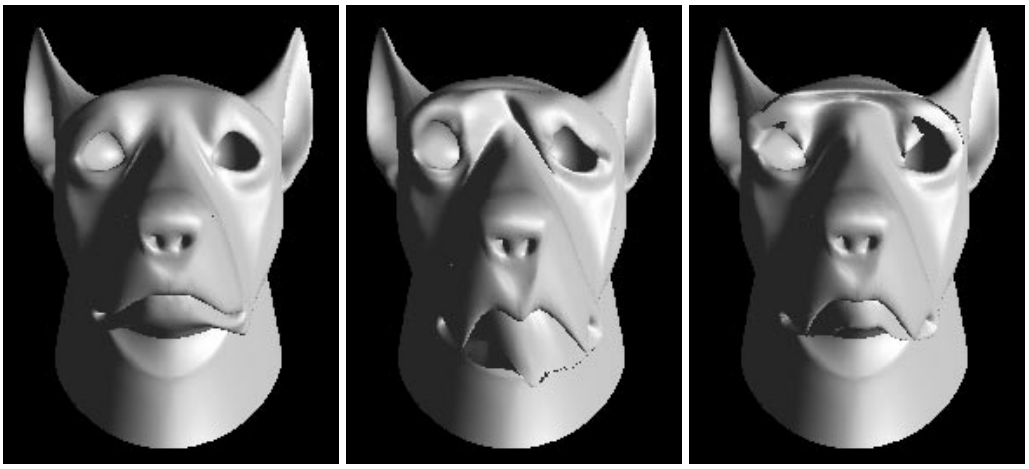

Figure 6.3: Attachments made at low level — neutral, frame 82, and frame 110

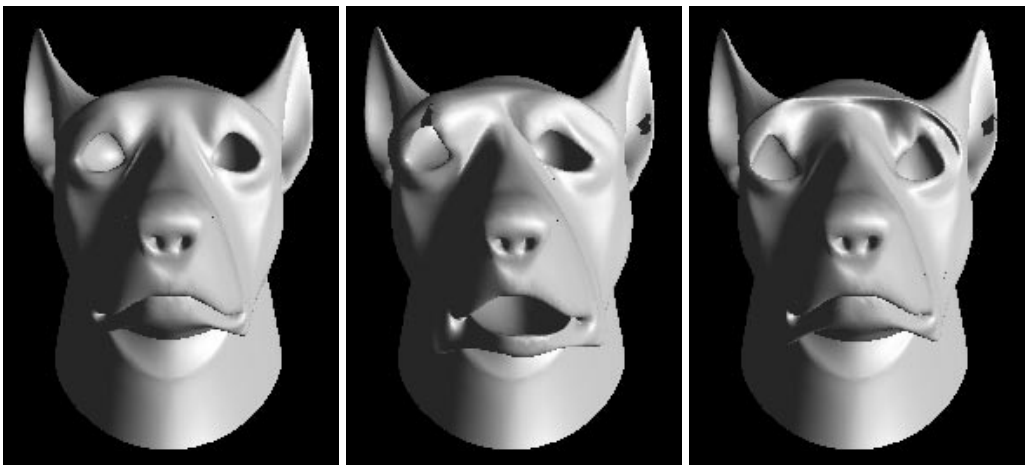

Figure 6.4: Attachments made at high level — neutral, frame 82, and frame 110

have to move in a straight line. Also, the control vertices can move independently within a given action. The drawback to using actions in Dragon is that they take longer to set up.

Consider FAP 4, "lower_t_midlip". This facial animation parameter corresponds to the top middle lip moving down. Its units are "MNS", or proportional to the distance between the mouth to the nose. Figure 6.5 shows the resulting image of the dog with FAP 4 values of -1000, 0, and 1000. Actual values for the FAP usually range from around -50 to 250, so poses as extreme as these will not normally be seen.

Figure 6.5: FAP 4, "lower_t_midlip" — -1000, 0, and 1000

Once the 16 actions for the 16 FAPs used in this data set were defined, a global action was created from the MPEG-4 data which drives the 16 FAPs which in turn move the associated control vertices. This global action can be used with any dragon animation which has those same 16 FAP actions defined. Figure 6.6 shows the neutral position, frame 6, and frame 94 of the dog model animated with MPEG-4 data. Figure 6.7 shows the neutral position, frame 6, and frame 94 of the dragon model animated with the same data.

The final images for the MPEG-4 data look reasonable because each of the individual FAPs was done in a reasonable manner. Care was taken to avoid very unnatural looking poses at each step.

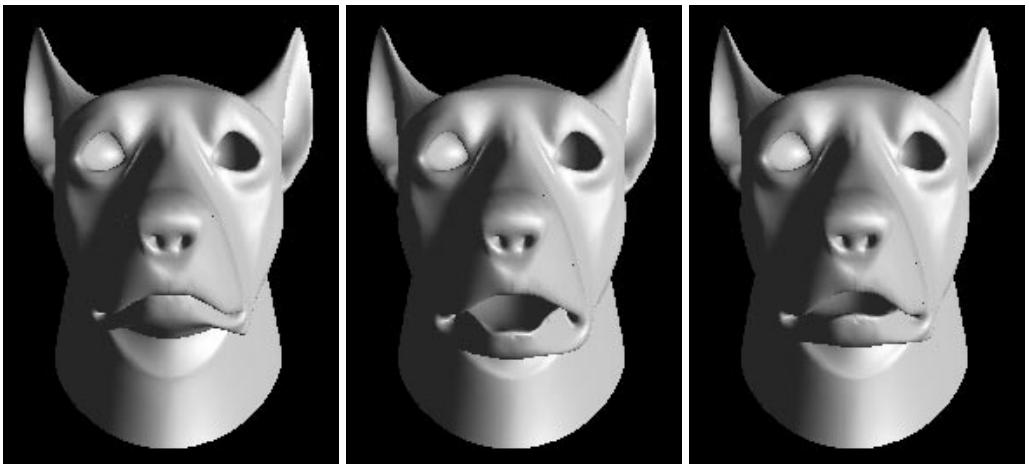

Figure 6.6: MPEG-4 dog — neutral, frame 6, and frame 94

Figure 6.7: MPEG-4 dragon — neutral, frame 6, and frame 94

Chapter 7

Conclusions

MPEG-4 FAPs offers efficiency over equivalent motion captured data at the cost of accuracy. It has been specifically designed to be as useful as possible with as few actions as are reasonable.

The facts that facial animation parameters have been carefully chosen and specified in relative units makes it intuitive for the animator to implement those parameters as actions in Dragon. In contrast, motion captured data offers no such relationships and so the animator must either implement the animations directly (which can lead to unnatural actions), create equivalent actions (of which there would be nearly two and a half times as many), or have some tool which automatically generates the actions based on the shape of the head.

Future work would be to create such a tool which would try to create the actions in a reasonable way based on the shape of the head. One idea is to morph between the two heads using the standard feature based metamorphosis algorithm[1] extended to 3D[13].

Bibliography

[1] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In Ed Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 35–42, August 1992.

[2] P. Ekman and W.V. Friesen. *Facial Action Coding System (Investigator's Guide)*. Consulting Psychologists Press, Inc., Palo Alto, California, USA, 1978.

[3] International Organisation for Standardisation. Mpeg–4 snhc vm. *URL: http://www.cselt.stet.it/mpeg/documents/snhcv20.htm*, April 1996.

[4] David R. Forsey. Creating facial actions with hierarchical splines. *URL: http://www.cs.ubc.ca/nest/imager/contributions/forsey/dragon/FaceTutorial.html*, April 1997.

[5] David R. Forsey and Richard H. Bartels. Hierarchical B-spline refinement. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 205–212, August 1988.

[6] P. Griffin and H. Noot. The fersa project for lip-sync animation. In *Proceedings Image'Com 93*, pages 111–116, Bordeaux, France, March 1993.

[7] V. Hall. Speech driven facial animation. Thesis, Curtin University of Technology, Perth, Australia, November 1992.

[8] David R. Hill, Andrew Pearce, and Brian Wyvill. Animating speech: an automated approach using speech synthesised by rules. *The Visual Computer*, 3(5):277–289, March 1988.

[9] D.R. Hill, A. Pearce, and B. Wyvill. Animating speech: An automated approach using speech synthesised by rules. *The Visual Computer*, 3:277–289, 1988.

[10] Prem Kalra, Angelo Mangili, Nadia Magnenat Thalmann, and Daniel Thalmann. 3D interactive free form deformations for facial expressions. In *COMPUGRAPHICS '91*, volume I, pages 129–141, 1991.

[11] F. Lavagetto and S. Curringa. Videophone coding based on 3d modeling of facial muscles. In *SPIE Visual Communications and Image Processing'92*, pages 1366–1374, Boston, USA, 1992.

[12] Yuencheng Lee, Demetri Terzopoulos, , and Keith Waters. Realistic modeling for facial animation. In Robert Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 55–62, August 1995.

[13] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy. Feature-based volume metamorphosis. In Robert Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 449–456, August 1995.

[14] J. Lewis. Automated lip-sync: Background and techniques. *The Journal of Visualization and Computer Animation*, 2:118–122, 1991.

[15] J. P. Lewis and F. I. Parke. Automated lip-synch and speech synthesis for character animation. In J. M. Carroll and P. P. Tanner, editors, *Proceedings of Human Factors in Computing Systems and Graphics Interface '87*, pages 143–147, April 1987.

[16] N. Magnenat-Thalmann, E. Primeau, and D. Thalmann. Abstract muscle action procedures for human face animation. *The Visual Computer*, 3(5):290–297, March 1988.

[17] N. Magnenat-Thalmann, E. Primeau, and D. Thalmann. Abstract muscle action procedures for human face animation. *The Visual Computer*, 3(5):290–297, 1988.

[18] S. Morishima and H. Harashima. A media conversion from speech to facial image for intelligent man-machine interface. *IEEE Journal on Selected Areas in Communications*, 9(4):594–599, May 1991.

[19] S. Panis, J. Cosmas, and C. Cock. Control of movement of a synthesised mouth from a text stream. In *Proceedings IEE 4th Int. Conf. on Image Processing and its Applications*, pages 266–269, Maastricht, The Netherlands, April 1992.

[20] F.I. Parke. Parameterized models for facial animation. *IEEE Computer Graphics & Applications*, pages 61–68, Nov. 1982.

[21] F.I. Parke. Techniques for facial animation. In N. Magnenat Thalmann and D. Thalmann, editors, *New Trends in Animation and Visualization*, chapter 16, pages 229–241. John Witley & Sons Ltd., 1991. (Include figure 16 and 17 on color pages).

[22] S.M. Platt and N.I. Badler. Animating facial expressions. *ACM Computer Graphics*, 15(3):245–252, Aug. 1981.

[23] Marcel Reinders. Bibliography on computer vision and facial animation. *URL: http://cosmos.kaist.ac.kr/pub/bibliographies/Graphics/facial.animation.html*, April 1997.

[24] E.A. Salzen. The construction of emotion from facial action. In A.W. Young and H.D. Ellis, editors, *Handbook of Research on Face Processing*, chapter Expression, pages 177–186. Elsevier Science Publishers B.V. (North-Holland), 1989.

[25] A. Saulnier, M. Viaud, and D. Geldreich. Real-time facial analysis and synthesis for televirtuality. In *Proceedings WIASIC'94 Workshop on Image Analysis and Synthesis in Image Coding*, page IP2, Germany, October 1994.

[26] D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation*, 1:73–80, 1990.

[27] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6):569–579, June 1993.

[28] K. Waters. A muscle model for animating three-dimensional facial expression. *ACM Computer Graphics*, 21(4):17–24, July 1987.

[29] K. Waters and T.M. Levergood. Decface: An automatic lip - synchronization algorithm for synthetic faces. Technical Report CRL 93/4, Digital Equipment Corporation, Cambridge Research Lab, September 1993.

[30] K. Waters and D. Terzopoulos. A physical model of facial tissue and muscle articulation. In *Proceedings First Conference on Visualization in Biomedical Computing (VBC'90)*, pages 77–82, Atlanta, GA, May 1990.

[31] K. Waters and D. Terzopoulos. Modelling and animating faces using scanned data. *The Journal of Visualization and Computer Animation*, 2:123–128, 1991.

[32] W.J. Welsh, A.D. Simons, R.A. Hutchinson, and S. Searby. A speech-driven 'talking-head' in real time. In *Proceedngs Picture Coding Symposium*, pages 7.6.1–7.6.2, Cambridge USA, March 16-28 1990.

[33] L. Williams. Performance-driven facial animation. *ACM SIGGRAPH '90*, 24(4):235–243, August 1990.

[34] J. F. S. Yau and N. D. Duffy. 3D facial animation using image samples. In N. Magnenat-Thalmann and D. Thalmann, editors, *New Trends in Computer Graphics (Proceedings of CG International '88)*, pages 64–73. Springer-Verlag, 1988.