An Evaluation of Tension within an Extensible Spline Testing Facility

Paul Ruest

CS-89-53

November, 1989

An Evaluation of Tension within an Extensible Spline Testing Facility

by

Paul Ruest

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Masters of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1989

©Paul Ruest 1989

Abstract

Cubic Beta2 splines and rational B-splines have received a great deal of attention in recent years. Both associate an extra parameter called tension with each control vertex which allows the curve to be pulled or pushed locally, relative to the control polygon, and independent of control vertex movement.

An obvious question from the design point of view is whether users actually find these tension parameters useful. Therefore an experiment was performed to explore the usefulness of tension when interactively designing curves. The experiment results argue for using a few additional control vertices in ordinary cubic B-spline curves instead of a tension spline.

The experiment was performed on a reworked version of an extensible spline testing software system that has been developed at the University of Waterloo Graphics lab.

Acknowledgements

Ian! Allen: software style support, Dr. John C. Beatty: supervisor, Dan Christensen: tension interface addition, Dr. William Cowan: colour input, Dr. Pierre Jolicoeur: experimental design support and faculty reader, Alan Paeth: Im toolkit expertise, Suzanne Potvin and Al Vermeulen: student readers, Dr. Grant Weddell: faculty reader, and funding by NSERC PGS scholarships and the Computer Graphics Laboratory through DEC, ITRC and URIF grants.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Always one to embrace man's avarice for efficiency, computer graphics has adopted splines with enthusiasm in the last decade. This is not simply a fad; there are industries to which ideal curve rendition and representation matter significantly, notably the automotive industry and the computer font industry.

Alternate curve representation methods exist, but do not challenge the spline in flexibility and efficiency. For example storing curves as a multitude of straight line segments is costly in space and the resulting curves are difficult to manipulate. Joining different size circular and elliptical arcs is more compact but insufficiently general. Additionally both "solutions" define an inferior emulation of what graphic artists have been doing for centuries – hardly the giant stride forward one expects from the computer industry.

Splines are a mathematically precise, space-efficient method for representing complex curves. The curves are continuous over large intervals and are easy to extend and shape as desired. Splines are based on combining polynomial curve segments to form a complete curve.

The splines of interest to this thesis are discussed in section 1.1: Spline Basics. It will be obvious from section 1.1 that there are a number of mathematical representations for splines, each offering different properties. It is for this reason that experiments are needed to determine which are easiest to learn and use.

This thesis is an extension from a first set of spline experiments performed by Eric George Bosch [Bosch 87]. Bosch's experiment studied the manipulation of curves defined by a series of points associated with the spline curve called *control vertices*; repositioning control vertices created a differently shaped curve. Subjects manipulated a *controlled* curve in one viewport, attempting to reproduce the shape of a generated *target* curve shown in an adjacent viewport. The spline technique by which this could be accomplished most quickly and accurately was considered to be the best. Bosch's study, involving B-spline, Bézier, C^2–Interpolating and Catmull-Rom splines, is summarized in section 1.2: Background. The reader is encouraged to familiarize himself with Bosch's work as this will facilitate understanding this document.

This thesis extends Bosch's work by studying tension: an attribute of some splines that allows the reshaping of a curve by means other than repositioning the control vertices. With tension a wider variety of curves can be represented. Hence, the current assumption is that the inclusion of tension in splines is a boon. However this assumption has not been verified. Thus a controlled experiment has been implemented evaluating whether tension is beneficial for spline curve manipulation. The new experiment, which deals with B-splines, Beta2 splines and rational B-splines, is summarized in section 1.3: New Experiment.

The remaining thesis chapters deal with two topics: the major changes in Bosch's code, resulting in the current experiment code, and the tension experiment itself. The prominent changes affecting the experiment system are in experiment control, the internal spline data representation and target curve generation.

Changes in experiment control stemmed from the software design goals of the current shape matching system. Chapter 2, Experiment Organization, describes the design goals and the software control required to run an experiment.

An important software change was the inclusion of the two tension splines, rational B-splines and Beta2 splines. These changes affected the spline representation and the curve manipulation interface. Internally crucial, the spline is now represented as an independent, abstracted spline object. Externally crucial, a tension interaction method has been devised and integrated within the current interface. These internal and external spline elements are discussed in Chapter 3: Computer Splines.

The final change was the use of real-life target curves. Details of the collecting process and the collection environment are discussed in Chapter 4: Target Curves.

Finally, Chapter 5, Tension Experiment, discusses what the tension experiment hoped to achieve, the experiment methodology and the final results.

1.1 Spline Basics

The three splines of concern to this thesis are: B-splines, Beta2 splines and rational B-splines. They all share terminology that needs to be defined for later use in the thesis. Note that only the terminology of interest to this thesis is presented; for more detail see [Bartels 87].

The rudimentary elements of a spline curve are shown in Figure 1.1. These are:

- The curve is a composite of a number of *segments*, shown here by alternating regular and dashed lines. The locations at which the segments connect are called *joints*.

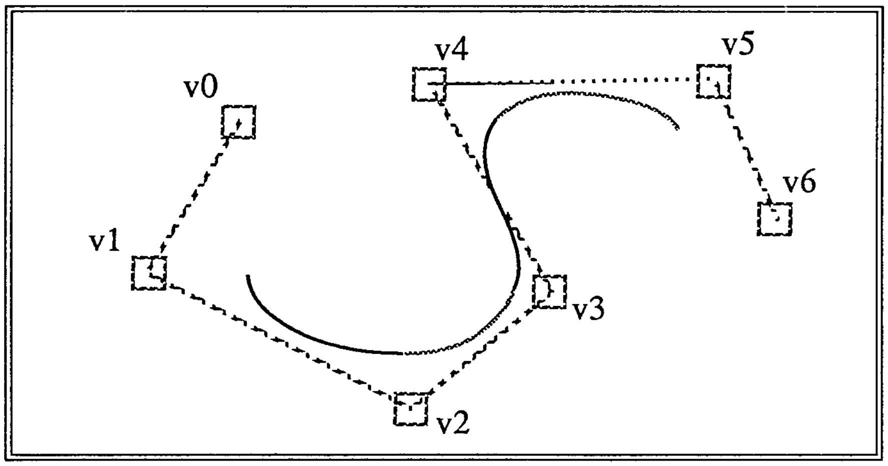


Figure 1.1: A Spline Curve.

- The curve is defined by positioning some number of *control vertices* (represented by square boxes), near which the curve passes. The control vertices can be repositioned to change the shape of the curve. For the three types of splines under consideration, moving a control vertex will cause only a portion of the curve to change, a property called *local control.* Some other spline types provide *global control*, for which repositioning a control vertex causes the entire curve to change shape.

- The control vertices are approximated in some order. This order is represented by connecting the control vertices with line segments to form what is called the *control polygon.* The *convex hull* of a set of vertices is the area within the outline of the control vertices [Bartels 87]. A curve contained within this outline is said to have the convex hull property.

- A certain amount of "smoothness" is required. Each segment, being a polynomial, is infinitely differentiable and therefore "infinitely smooth". Smoothness at the joints is achieved by requiring that the abutting curves have some number of matching derivatives. This requirement is known as *parametric continuity.* When the first and second derivatives are required to agree, the resulting curves are said to be C^2 *continuous.*

Mathematically the three splines under discussion can be represented by means of blending functions. This is mathematically shown as:

$$Q(u) = (X(u), Y(u)) = \left(\textstyle\sum_{i=0}^{m} V_{xi}B_i(u),\ \sum_{i=0}^{m} V_{yi}B_i(u) \right)$$
$$\text{for} \quad u \in [u_3 \leq u_i \leq u_{m+4})$$

where, with the self imposed restriction to cubic polynomials:

- $u_0 < u_1 < u_2 < \ldots < u_m$ is a sequence of of equally spaced *knots*, and the independent parameter u varies from u_3 to u_{m+4} as it traces out the curve. Qualitatively u represents distance along the curve, although it is not arc length.
- $Q(u)$ represents the entire spline curve,
- V_{xi}, V_{yi} are the x and y components of the control vertices, and u is non-zero for $u \in (u_i, u_{i+4})$.
- $B_i(u)$ is the basis function associated with the control vertex components V_{xi} and V_{yi}.

Figure 1.2 illustrates what is meant by blending functions. Each point on $Y(u)$ is a "blend" of control vertices in that it is a scaled sum of control vertices where the scale factors are exactly the $B_i(u)$.

The individual basis functions are non zero over four intervals, ($u0$ to $u4$ in Figure 1.3), and are therefore comprised of four cubic polynomial sections ($s_i(u)$). These

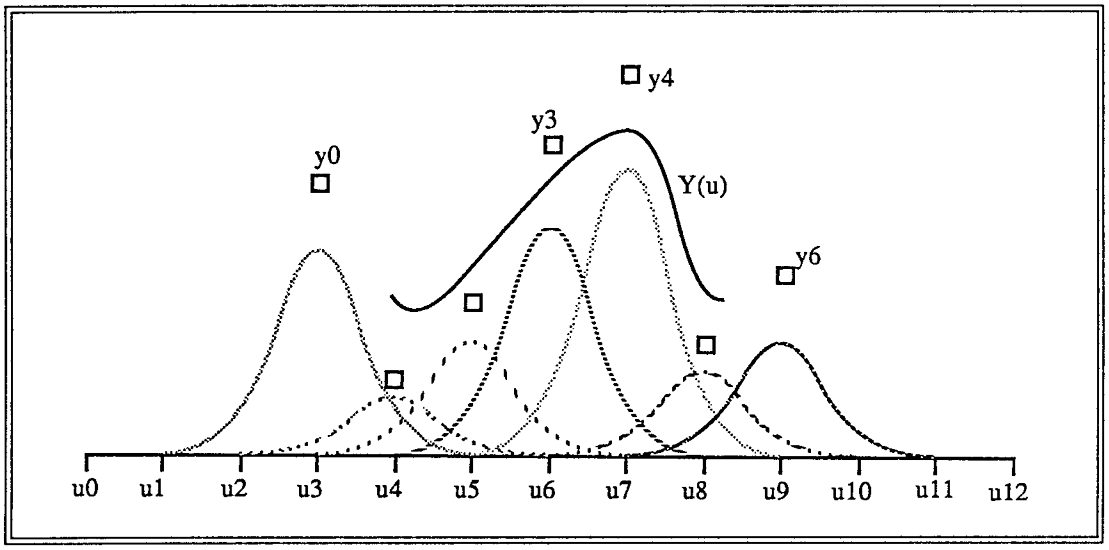


Figure 1.2: Summed Basis Functions (Blending).

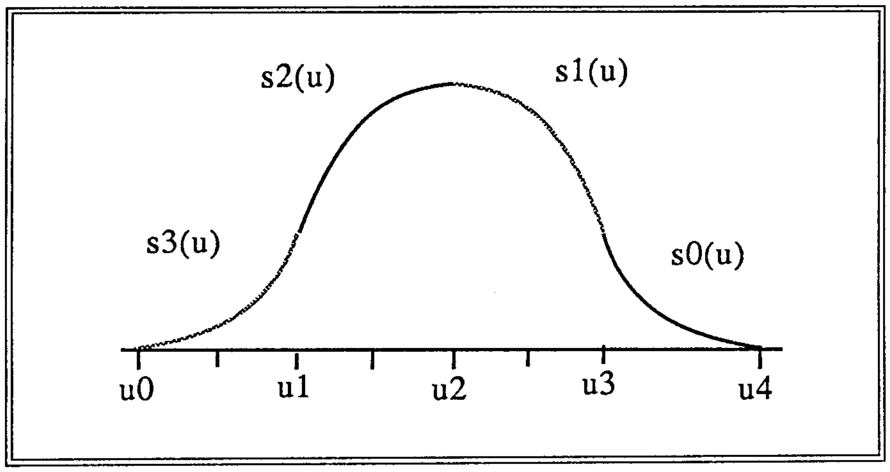


Figure 1.3: Spline Basis Function.

polynomials are defined so as to yield the spline properties discussed above: i.e. C^2 continuity for the B-spline and Rational B-spline.

The rational B-splines and Beta2 splines have an extra value associated with each control vertex, which is referred to generically in this thesis as a *tension* value. Tension is beneficial in three ways:

1. It allows reshaping of the curve without repositioning the control vertices.
2. It allows a means of locally pulling the curve towards the control polygon or of pushing it away from the control polygon.
3. It provides a means of realizing sharp corners.

Negative tension has the effect of pushing the curve away from the control polygon, possibly violating the convex hull property.

The inclusion of tension changes the spline definition slightly. In the case of the rational B-spline (introduced in Versprille and described in [Roy 88] and elsewhere) the tension value, weights labeled w, are associated with each basis function. The rational B-spline derives its name from the fact that its basis functions $B_i^*(u)$ are formed from the division of two splines:

$$B_i^*(u) = \frac{w_i B_i(u)}{\sum_{j=i-3}^{i} w_j B_j(u)}$$

The shape of the rational B-spline curve is affected by the relative tension values for each curve segment. If one of the w values is larger than the others then the associated basis function has a greater affect on the spline curve. The curve is thus pulled towards control vertices with higher tension values than surrounding control vertices. Hence from a user's point of view each weight appears to be associated with a unique control vertex.

The Beta2 spline provides a similar control and is a special case of the more general Beta splines (see [Bartels 87]). Beta splines are a generalization of B-splines and are formed by demanding *geometric continuity* instead of parametric continuity at the joints.

Geometric continuity demands equal values for position, normalized first derivative, and curvature vector. These constitute a partial relaxation of parametric continuity, in the process of which the simple equality of parametric derivitives is replaced by more complex equations that are functions of two independent variables β_1 and β_2. The coefficients of the segment polynomials also become expressions in β_1 and β_2. β_2 provides the tension-like behaviour described earlier, and for the purpose of this thesis, β_1 is set to one (which results in first degree parametric continuity at the joints). This had the added value of simplifying the segment polynomials which, for the first segment defined, become:

$$\begin{aligned}
s_3(u) = {} & \tfrac{1}{\delta 1}(2(\beta_{2,5}+4)u^3) \\
s_2(u) = {} & \tfrac{-1}{\delta 1 \delta 2}2(\beta_{2,4}+4)[\beta_{2,3}\beta_{2,4}\beta_{2,5}+8\beta_{2,3}\beta_{2,5}+8\beta_{2,3}\beta_{2,4} \\
& \quad +3\beta_{2,4}+4\beta_{2,5}+44\beta_{2,3}+24\beta_{2,4}+28\beta_{2,5}+144]u^3 \\
& \quad +\tfrac{1}{\delta 2}(\beta_{2,4}+4)[3(\beta_{2,3}+2)u^2+6u+2] \\
s_1(u) = {} & \tfrac{-1}{\delta 2 \delta 3}2(\beta_{2,3}+4)[\beta_{2,2}\beta_{2,3}\beta_{2,4}+8\beta_{2,2}\beta_{2,4}+3\beta_{2,2}\beta_{2,3} \\
& \quad +8\beta_{2,3}\beta_{2,4}+28\beta_{2,2}+24\beta_{2,3}+44\beta_{2,4}+144](1-u)^3 \\
& \quad +\tfrac{1}{\delta 3}(\beta_{2,3}+4)[3(\beta_{2,4}+2)(1-u)^2+6(1-u)+2] \\
s_0(u) = {} & \tfrac{1}{\delta 3}(2(\beta_{2,2}+4)(1-u)^3)
\end{aligned}$$

where

$$\begin{aligned}
\delta 1 = {} & \beta_{2,4}\beta_{2,5}+8\beta_{2,4}+8\beta_{2,5}+48 \\
\delta 2 = {} & \beta_{2,3}\beta_{2,4}+8\beta_{2,3}+8\beta_{2,4}+48 \\
\delta 3 = {} & \beta_{2,2}\beta_{2,3}+8\beta_{2,2}+8\beta_{2,3}+48
\end{aligned}$$

Thus the β_1 will have the value one for the remainder of the thesis, while the β_2 will be permitted to vary to provide the desired tension effect. Thus the spline studied is a subset of the Beta spline, and is aptly called the Beta2 spline.

1.2 Background

The first curve matching experiment, that is to be described in this section, was performed by Eric Bosch [Bosch 87]. The goal of the experiment was to test ease, accuracy and quickness in manipulating four spline types: B-spline, Catmull-Rom (called Cardinal in Bosch's work), C^2–Interpolating, and Bézier.

The experiment is comprised of *sessions* and *trials.* A session is a unit of time encompassing a number of trials. A trial is denoted as the matching of one *controlled* curve to one *target* curve. A controlled curve is a spline that the subject is able to manipulate. This spline is changed to match the target curve, a spline which the subject cannot manipulate. Each trial begins with an identically initialized controlled curve and a different target curve.

At the start of the first session the subject is provided with computer guided instructions, which explain in detail how to perform and evaluate a trial. During the instructional phase excerpts of an actual trial are presented to a subject. By the end of the instructions one complete match will have been attempted.

The interface for each trial consists of three successive screens: the start screen, the match screen and the rating screen. The initial trial screen (Figure 1.4) consists of two empty viewports and a start button. To begin a trial the start box is selected.

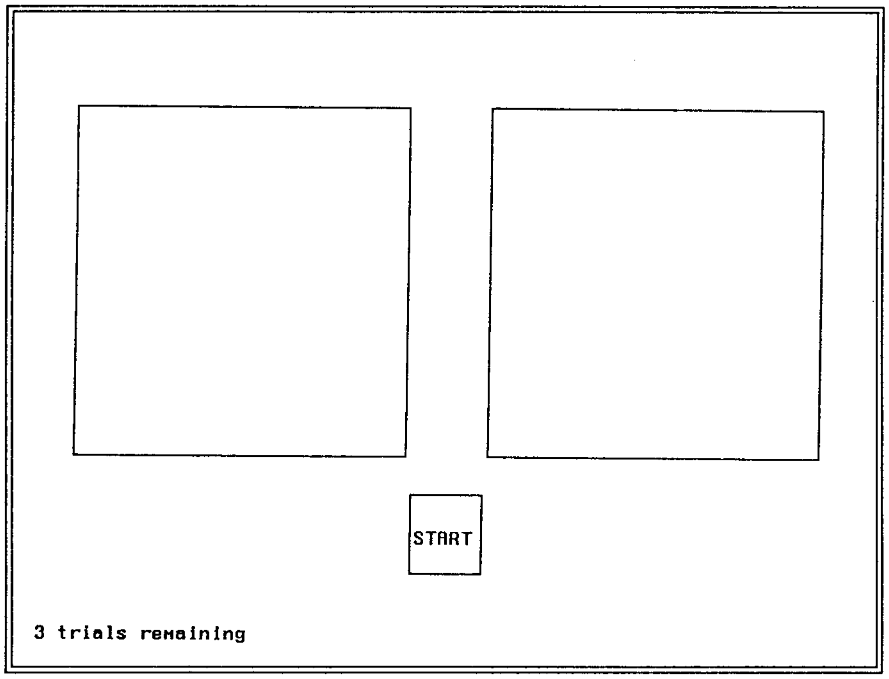


Figure 1.4: Start Screen.

The second screen is depicted in Figure 1.5. It contains the target curve drawn in the square viewport on the left side of the screen, the controlled curve in the square viewport on the right side of the screen, and a stop button replacing the previous start button. The square boxes within the right viewport represent control vertices. These vertices are connected by straight lines, on the display, to form the control polygon. The viewport boundaries and text are drawn in white, the target and controlled curves are full intensity green and the control polygon is a medium grey. A successfully selected control vertex is drawn in full intensity red.

At the beginning of a match the control vertices for the controlled curve are equally spaced from the lower left to the upper right of the right hand viewport. The control vertices can then be repositioned by the subject to alter the controlled curve. The controlled curve can be translated if the mouse icon is initially in the controlled view-

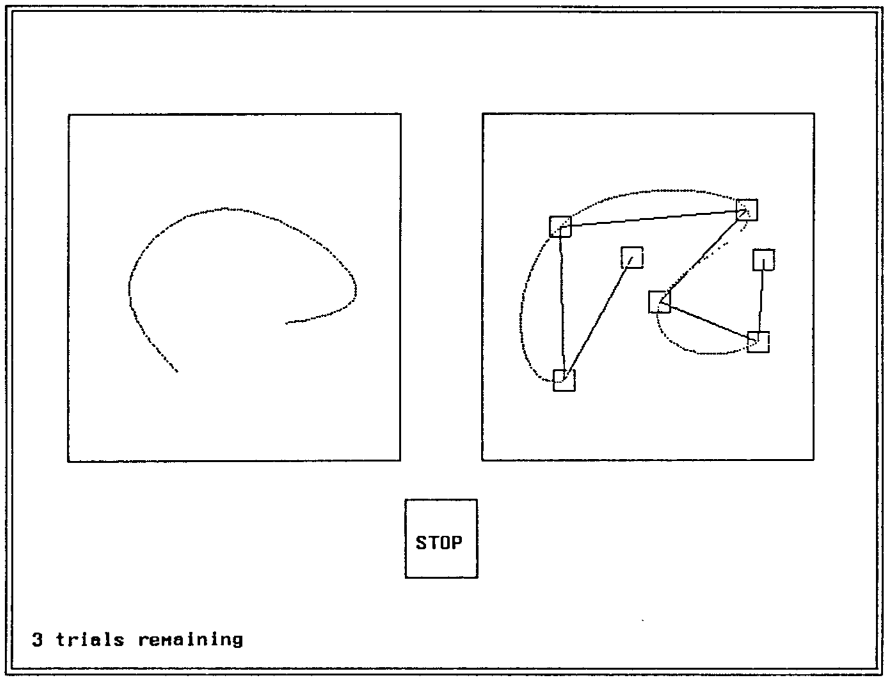


Figure 1.5: Match Screen.

port but not in a control vertex. Then the curve is translated by depressing a mouse button and moving the mouse icon the desired distance of translation. Moreover the control polygon visibility can be toggled by a mouse click outside the controlled viewport. The stop button is selected when the subject believes that the match is complete or cannot be improved.

The final screen is the rating screen shown in Figure 1.6. The control polygon is removed and the subject is no longer able to alter the controlled curve. The stop button is replaced by a rating scale which the subject uses to state their opinion of the match quality. A rating square containing the mouse cursor is outlined in white and is coloured red when selected.

All trials are recorded by saving all *spline manipulation actions.* These consist of user actions that change the controlled curve in any way; e.g. moving a control

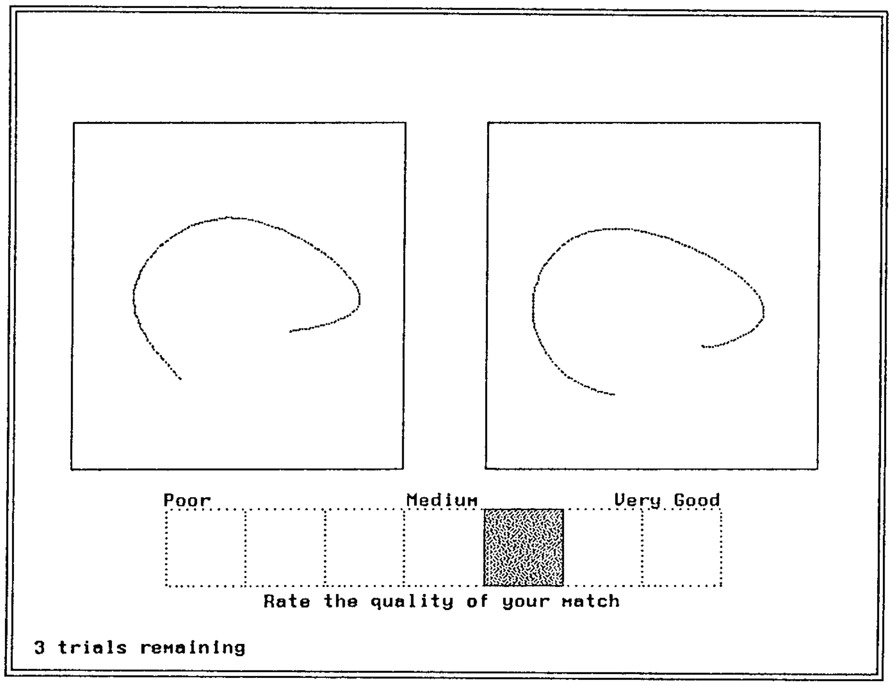


Figure 1.6: Rating Screen.

vertex or toggling the control polygon. The manipulation information is saved in a text format by using two character keywords to represent different subject actions. The keywords are followed by cursor coordinates, curve vertex data, time stamps and other event related information. All time stamps are recorded in units of one sixtieth of a second.

The saved information is used to evaluate and plot an objective match closeness as well as to replay the subject's actions at better than 95 percent real time. The objective evaluation method uses the angles made by equal length parameter steps along the two curves; the closer the corresponding angles between curves, the better the match.

Two experiments were run, with this system, each with 16 subjects performing 64 trials for a total of 1024 trials. In each experiment subjects had four sessions in which each session had four practice trials and 16 experimental trials. In the first experiment target curves were chosen randomly from the intersection space of the four spline types. Note that even though each spline type generates a mathematically different set of splines there is a non-empty intersection space. To lie in the intersection space target curves must be constrained to have C^2 continuity. In the second experiment target curves were chosen from the space of curves generated by each individual controlled spline type and each spline type was tested against target curves belonging to the same spline space.

The experiment results for both where analysed using an analysis of variance on the means to determine statistical significance. The main component of the analysis is a standard F-test [Ostle 75].

The results of the first experiment, Table 1.1, indicate that the B-spline is superior due to its high mean objective rating and low mean time. The results however might be biased towards the spline types with inherent C^2 continuity, like the B-spline, because of the C^2 continuity constraint for target curves.

Method	Mean Objective Rating min = 0.0, max = 1.0	Mean Time seconds	Mean Subjective Rating min = 1, max = 7
B-spline	0.8900	105.900	5.2300
Bézier	0.8300	127.300	5.0600
Cardinal	0.7600	118.100	4.9500
Interpolating	0.6500	130.100	4.6900
$F(3, 45)$	21.0690	3.896	5.7570
$\rho <$	0.0001	0.0148	0.0021

Table 1.1: First Experiment Results.

The second experiment results are less conclusive since only Mean Objective Rating is statistically significant, $\rho < 0.05$. However the first and last ranking splines do not change position as only the the Bézier and Cardinal splines change rankings. As such, the B-spline still fares the best in the only statistically significant result. The results of this experiment are summarized in Table 1.2.

Method	Mean Objective Rating	Mean Time	Mean Subjective Rating
	min = 0.0, max = 1.0	seconds	min = 1, max = 7
B-spline	0.8900	130.1000	4.9100
Bézier	0.6900	127.0000	4.7000
Cardinal	0.8300	112.9000	4.9200
Interpolating	0.6400	139.2000	4.8000
$F(\ 3,\ 45\ )$	36.9380	2.3430	1.0140
$\rho <$	0.0001	0.0858	0.3956

Table 1.2: Second Experiment Results.

1.2.1 Code

Bosch's original *shape matching system* code, written in C and Unix System *V* shell script, is stored in several subdirectories. The relevant code resides within five programs: `Proto` controls a session, `Help` provides experiment instruction, `Targets` generates target curves in the intersection space of the controlled spline types, `Targets2` generates target curves in any single spline space and `Playback` displays and evaluates the data generated by `Proto`. Each program is kept in a single directory with its own source, bin and data subdirectories. General data files are found in diverse top level directories. In this subsection each program is introduced and on certain implementation weaknesses are commented on.

`Proto` is the core program since its routines and type definitions are referenced by all of the other programs. `Proto` is responsible for controlling a session. To execute, it requires an input file containing target curves and command line parameters. It outputs experiment data that is later used by `Playback`.

The `Help` program uses a button interface to guide the novice user though a series of informational text pages and demos. The only input file required contains the instruction text and embedded control words for demos. The demos are shown in a separate pop-up window for subject viewing and practice. The possible demos are:

- showing the initial match screen,
- having the user move control points,
- having the user move the entire curve and toggle the control polygon,
- having the user try the rating scale and
- having the user try one complete match.

`Targets` generates target curves in the intersection space of experiment controlled spline types. An identical spline curve, for different control vertex placements, is displayed in four different viewports corresponding to the controlled spline types. The experimenter either rejects or accepts the target curve. In the former situation, a new curve is randomly created; in the latter situation, the curve is appended to an output file. This output file is one of many `Proto` target curve input files.

`Targets2` is identical to the `Targets` program except that it generates curves from a specific spline space.

`Playback` uses the playback information to display and evaluate subject trials. However, only spline manipulation actions can be viewed. These are: moving the control points, translating the curve, toggling the control polygon and choosing a rating selection. While the subject's actions are replayed in one window, the evaluated match closeness is plotted in a second window and the corresponding numerical rating is displayed in a third window.

Complete experiment control is accomplished with shell scripts which use a series of scratch files for calculating the spline type/target curve combination to employ for a given subject/session.

The process of porting Bosch's code to a new Iris and incorporating experiment changes exposed the following problems:

- Only one of the six programs has any comments.
- The programs lack coding consistency in that different routines and data types are used to accomplish identical tasks. Moreover, the coding style for each program is noticeably different.
- The programs are not portable. For example, all windows and viewports are hard coded by pixel address. This makes porting impossible, even to the same machine with a different monitor size. Additionally, files and programs are referenced through absolute hard coded pathnames.

  A slightly less obvious portability weakness stems from an operating system dependency. The Iris 2400 experiment platform expects programs to use the entire screen and the complete processing power of the graphics subsystem. Changing to a multi-window, multi-program operating system requires adding refresh, resize, and reposition capabilities.
- There is a distinct lack of "information hiding" due to an inordinate number of global variables. The global variables are not only used between modules in a single program directory but between modules in different program directories.

- There is a weakness in data abstraction due to poorly structured data types. This resulted in many unnecessary data objects, which made for awkward parameter passing.

Overall Bosch's original shape match system underwent a major rewrite to provide not only a more general and flexible structure but also a testbed that would include both Bosch's experiment and the current experiment.

The current experiment system and the experiment conducted with it are now introduced in light of the previous experiment. This is a brief overview of the new experiment interface and code organization. Greater depth is provided in later chapters.

1.3 The New Experiment

The *tension experiment* followed the Bosch's original experiment. Its goal is to compare the ease of manipulation for splines with tension against splines without tension. The two tension spline types considered are the rational B-spline and the Beta2 spline, while the non-tension spline types considered are the B-spline with seven control vertices and the B-spline with ten control vertices. Target curves are no longer randomly generated mathematically, but are instead digitized from real objects. More precisely, they are sampled from a large-scale set of Roman letters, and are comprised both of sharp cornered *cusp* curves, and of smooth, *non-cusp* curves to provide an appropriate testing ground for tension splines.

The experiment interface remains similar to the original experiment in order to provide backward compatibility. As before there are three screens: the start, match, and rating screens. Within each screen the two viewports have been combined into a single *match window*. The target and controlled curves are thus overlayed, providing more feedback for precision curve matching (Figure 1.7). The change provides a different means of objective evaluation for match closeness than did Bosch's experiment and offers an alternative paradigm for curve creation.

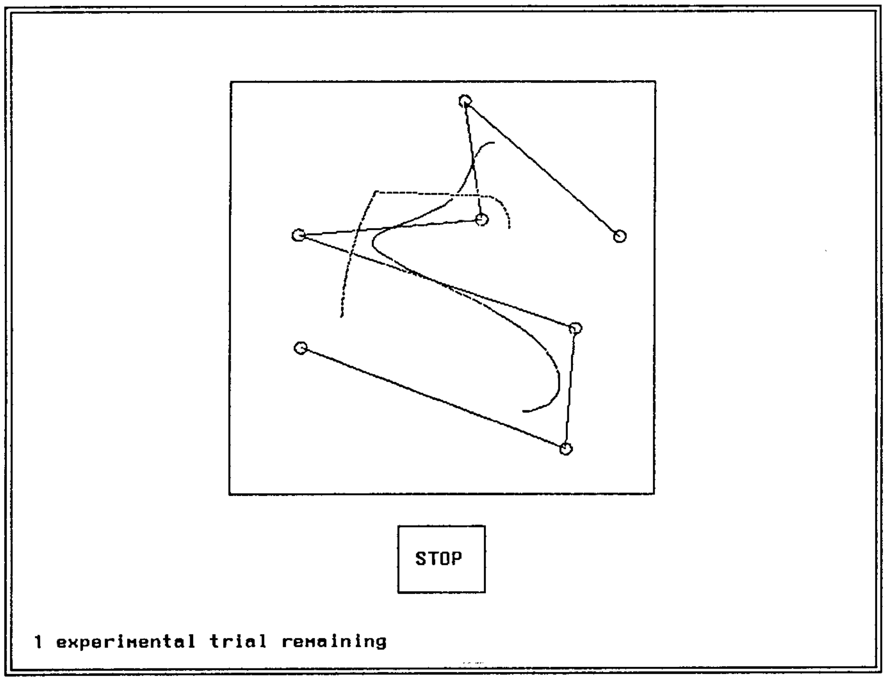


Figure 1.7: New Match Screen.

Due to the single match viewport, curve translation has been changed to affect both target and controlled spline curves simultaneously. Moreover, selecting a control vertex causes the remainder of the control polygon to disappear. This allows a better view of the changes occurring in the controlled curve. The use of tension splines also requires a slight modification to the control polygon. Finally, circular control vertices are used for tension splines, rather than the square boxes used for B-spline curves, so as to remind the user that tension is available.

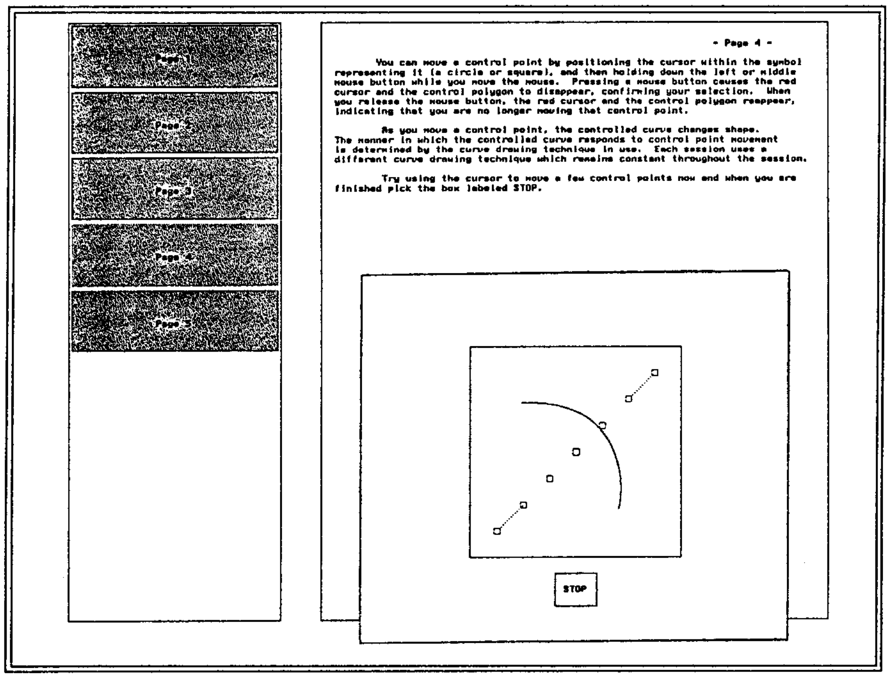
Another minor change is that colours are now used to distinguish the target and controlled curves and to indicate match closeness. The target curve is initially coloured brown and the controlled curve is initially coloured red. The portion of the controlled curve within one pixel of the target curve is coloured a bluish grey. An exact overlay is denoted by the colour green. All control vertices are shaded grey and dimmed upon selection. The use of red and green are meant as intuitive keys; red: bad, green: good. Moreover the subject's attention is directed to the key match elements through the use of these high colours.

1.3.1 Code

The tension experiment code is written in C and Unix System *V* shell script with a smattering of postscript, awk and lex. There are four principle programs. The `CurveMatch` program is responsible for running a single trial. `Help` again guides a user through a series of instructional text pages and demos. `Setup` creates a file specifying the experiment for each subject. `Control` runs the experiment using the information produced by `Setup`.

The `CurveMatch` program (previously `Proto`) is still the heart of the system, though reduced in functionality. It provides complete trial control. It runs a single trial, provides playback capability for a trial and performs objective match evaluations. As the subject performs a trial, manipulation information consisting of Iris queue events and time stamps is saved. The current time stamps are in units of 10,000'ths of a second. In this manner every cursor move and mouse click is retained for future analysis. A separate playback program is not needed since the queue events can be replayed through `CurveMatch`. At the end of a subject's trial the objective match evaluation is performed. The evaluation is a simple matter of tallying the different coloured screen pixels.

The modified **Help** interface is presented in Figure 1.8. The leftmost window contains page indices and the rightmost window contains the instructional text pages. Each index is associated with a single page of text information. Additional changes to **Help** include a modification to the text information content (Appendix A) and the inclusion of a tension demonstration.



- Page 4 -

You can move a control point by positioning the cursor within the symbol representing it (a circle or square), and then holding down the left or middle mouse button while you move the mouse. Pressing a mouse button causes the red cursor and the control polygon to disappear, confirming your selection. When you release the mouse button, the red cursor and the control polygon reappear, indicating that you are no longer moving that control point.

As you move a control point, the controlled curve changes shape. The manner in which the controlled curve responds to control point movement is determined by the curve drawing technique in use. Each session uses a different curve drawing technique which remains constant throughout the session.

Try using the cursor to move a few control points now and when you are finished pick the box labeled STOP.

Figure 1.8: Help Screen.

The third program, `Control`, is the experiment administrator. It keeps track of the subject files and experiment-related files. It also runs all of the sessions according to a set of experiment specification files. `Control` is now written in C rather than shell script to gain control over the current Iris windowing system and to speed up execution.

Setup generates the complete experiment specification needed by `Control`. The output specification, one file for each subject, is generated from a concise trial and session experiment description file.

The coding style for the above programs was heavily influenced by the corrective changes necessitated by Bosch's main coding problem: lack of generality. These corrective changes can be grouped into three main areas: data abstraction, parameter files, and directory structure.

The data abstraction changes required extensive modifications within the spline data definition and the screen data structure. The new spline and screen object models provide consistent and simple function parameters, eliminating the need for global variables. The spline object permits a consolidation of all spline dependent code into one directory while the screen object provides a generic definition used consistently across all programs.

A second generalization was the use of parameter files as a means of program initialization. The parameter files contain screen layout information, with windows specified as fractions of a canonical, 0 to 1 space and viewports now specified as fractions of the main window. The use of parameter files allows reconfiguration of the experiment screen layout by changing fractions in a text file. This method replaces the previous hard-coded screen dependent information.

The last modification to improve generality was a reworking of the directory structure. It is now a self contained relative structure with a top level called `match` (Figure 1.9). Under match are three main divisions: `files`, `source`, and `upkeep`. The `upkeep` directory contains the experiment code maintenance software, e.g. shell scripts for tape backup. The `source` directory groups all source code into separate directories, four of which correspond to the aforementioned main programs. The other directories under `source` are:

`analysis`: All the paraphernalia used to analyze the results produced by the shape match experiment is contained within this directory.

`bin`: This directory holds all of the experiment related executable code. This includes shell scripts, postscript shells and C binaries.

`bitcurves`: This directory contains the different C code and shell script based routines that are used to transform real-life curves into the expected shape match system input format.

`documentation`: The subject information sheets and consent forms are contained within this directory. These documents are in the form of troff source.

`generics`: This directory contains all the source code that is general enough to be used by any of the shape matching programs and which is not specific to any single program. For example it includes all spline dependent code, the object specifications and general constants.

`gettarget`: This directory holds the source of the ported and augmented `Targets2` program. This program, now named `GetTarget`, is described in Appendix B.

`lib`: This library directory comprises all of the compiled shape match system routines.

The `files` directory contains all non-executable and non-source files. This includes input files for the experiment and data files generated by the experiment. There are five subdirectories.

`curves`: All of the input curves for an experiment are kept in this directory.

`playback`: All user motions are saved in a one file per one trial basis. The file names are keyed with the subject identification number, the session number and the trial number.

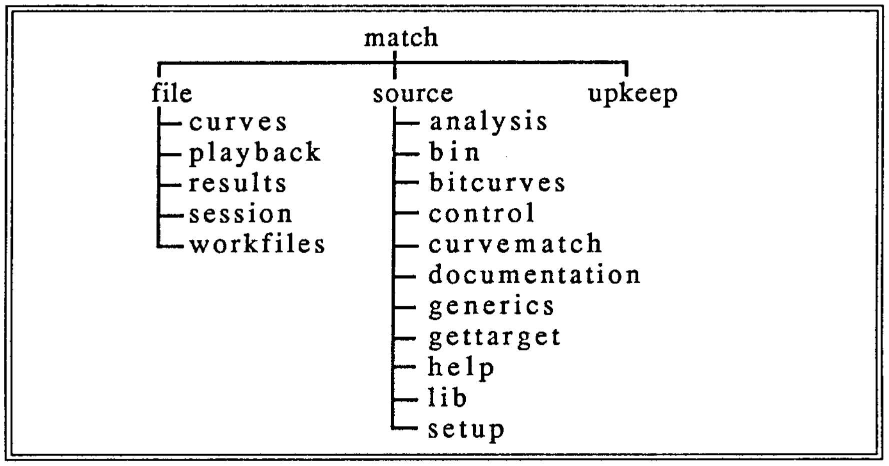


Figure 1.9: Directory Structure.

`results`: All of the experiment results are saved within this directory. This includes objective and subjective evaluation and timing information.

`session`: The experiment specification files are kept within this directory. These files are empty at the end of the experiment as their content is retrieved to control the experiment.

`workfiles`: This is a general directory that contains the miscellaneous files needed by the various programs. This includes the input parameter files for all the different programs.

The top level `source` and `file` directories will be used as reference points when describing the location of code and important files in the remainder of the thesis. Overall the new directory structure removes the necessity for absolute path names and gives a clear, concise structure to the shape matching software.

Chapter 2

System Organization

The shape matching system organization and design were influenced by interface requirements, code maintenance considerations, and the structure of the experiment.

The interface is designed to be as simple and intuitive as possible since the subjects of the shape match system are not expected to have previous computer experience. This is reflected in the use of simple controls for manipulating splines.

Good software programming practices have been, and must continue to be, adhered to. This is emphasized because new experimenters often start with little initial knowledge of the code or experiment, a direct result of the high turnover in the university graduate environment.

During an experiment the shape matching system progresses through several stages. These stages, and the corresponding programs, are:

1. experiment definition and setup (`Setup`),
2. running the experiment (`Control`),
    (a) experiment instructions and demonstrations (`Help`),

(b) a number of sessions (`Control`),

i. a number of trials

ii. producing trial data (`CurveMatch`),

3. experiment results and analysis.

This last stage is handled by the statistical tools provided in the `analysis` directory. The other stages and their interactions are presented in Figure 2.1.

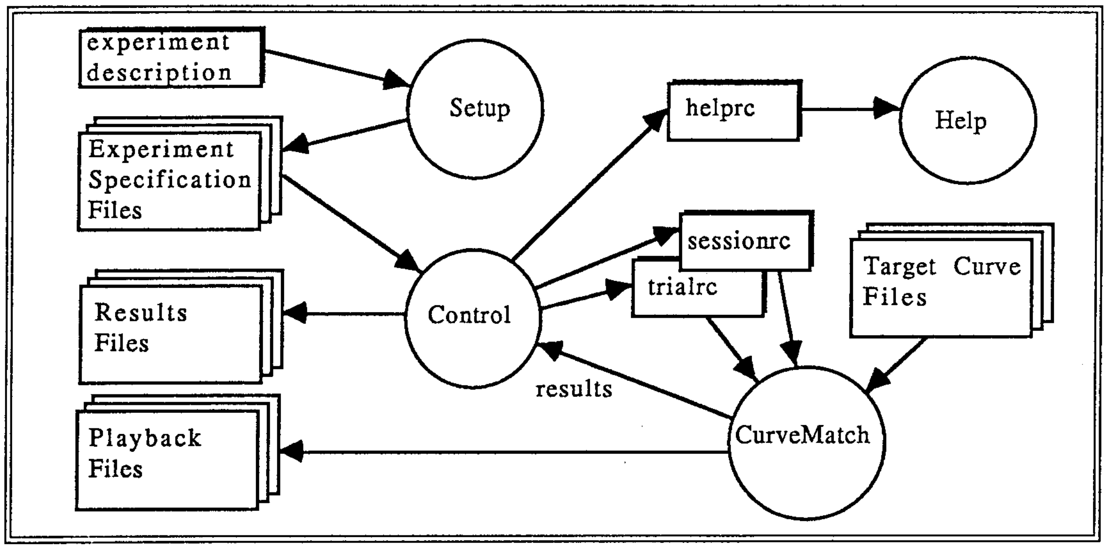


Figure 2.1: Organization Data Flow.

The three programs that orchestrate the running of a spline experiment, `Setup`, `Control` and `CurveMatch`, are presented in the remainder of this chapter.

2.1 Setup

The `Setup` program provides a complete experiment specification through the use of a robust experiment definition syntax. Its purpose is to transform a brief textual experiment description into a complete experiment specification.

The initial experiment description is stored in a text file created by the experimenter (Figure 2.2, see Appendix C for an example of the complete file). It supplies the answers to the basic questions of: how many subjects, how many sessions, how many trials, and what curves should the subject be using when?

```
subject 01 bspline-7-(AC) rationalb-7-(BD) beta-7-(AD) bspline-10-(BC)
subject 02 bspline-7-(AC) rationalb-7-(BD) bspline-10-(BC) beta-7-(AD)
subject 03 rationalb-7-(BD) bspline-7-(AC) beta-7-(AD) bspline-10-(BC)
                                 .
                                 .
                                 .
subject 24 bspline-10-(BD) beta-7-(AD) rationalb-7-(AC) bspline-7-(BC)
practice 1
experimental 10
```

Figure 2.2: Experiment Description.

In the file there is one line for each subject that is to take part in the experiment. The `subject` is followed by a line index followed by session descriptions; in this case there are four session descriptions. The session descriptions are of the format `a-b-(c)`, where `a` is the controlled spline type, `b` is the number of control vertices for the controlled spline, and `c` is a list of target curve set identifiers; The target *curve sets* are experiment input curves grouped and identified by a single capital letter.

The last two lines in Figure 2.2 indicate how many target curves are in each of the target curve sets. The "practice" line indicates that 1 practice curve should be used from every possible curve set: in this case one from each of `A`, `B`, `C`, `D`. The "experimental" line indicates how many experimental target curves are to be found in each of the sets indicated for the specific session. For example: the subject performing the first description line will have four sessions. In the first session the subject will manipulate a B-spline curve having seven control vertices. There will be four practice curves and 20 experimental curves, 10 from curve set `A` and 10 from curve set `C`.

The input experiment description file is based upon a few simple assumptions:

- Each subject's session has only one control spline type with an unchanging number of control vertices. For example, subject x might only manipulate B-spline controlled curves with seven control vertices during session y.
- A single session may consist of any number of curve sets.
- A curve set references a number of target curves, practice curves, and a help curve. These curves have been previously grouped by the experimenter according to the convention of using the curve set letter as an index to the individual target curves. Using the above example the target curves in set `A` consist of the experimental target curves `trial_A_01` to `trial_A_10`, the practice curve `practice_A_01` and the help curve `help_A`. These target curve files are kept in the `files/curves` directory. Each file contains a single target curve in a format that will be described later in Section 4.2: Processing Curves.

The other purpose of the input specification file is to facilitate arrangement of the target and controlled curves so that the experiment is not biased for or against any spline technique. How the experiment administrator arranges the splines is explained in Section 5.3: Blocking.

`Setup`'s main function is to randomly select the target curves from the target curve sets. This is further described in section 5.3: Blocking. `Setup` will output a series of text files, one for each subject, that will exactly specify each session and trial. An example of an output file is presented in Figure 2.3. The output text files are used as input to Control and explained in the next section.

2.2 The Session Driver: Control

The Control program is primarily a session driver. It handles running a complete session for each subject by calling the Help and CurveMatch programs, by interacting with subjects through a series of prompts, and by guiding the majority of the experimental data flow (Figure 2.1).

Control reads a file produced by Setup and generates the input parameter files required by Help and CurveMatch. The input file is organized by sessions as presented in Figure 2.3. See Appendix C for a more complete example of this session file.

The second through seventh lines are session control information that is currently embedded in Setup. The "mainwindow" line specifies the x and y coordinates of the experiment window relative to the Iris screen size. The viewport specifications, "targetview", ..., "ratingview", specify viewport sizes relative to the main window size. The sixth and seventh lines, also hardcoded in Setup, specify the location of the Help text file and the controlled and target splines to be used in the Help demos[1].

The following five lines describe the number of practice trials and the practice spline related information. The controlled spline type, its number of control points and target curve are specified for each practice trial. The information for four practice trials is presented in lines eight to twelve. The input files are from the files/session and files/workfiles directories. However, since all the examples assume that programs execute from the source/bin directory some indirection is needed to access

[1]The hardcoded information is generated by the Setup routine SeSessionPr().

```
session 1
   mainwindow 0.0 1.0 0.0 1.0
   targetview 0.25 0.75 0.25 0.875
   controlview 0.25 0.75 0.25 0.875
   startstopview 0.45 0.55 0.1 0.2
   ratingview 0.3 0.7 0.1 0.2
   helpfile ../../files/workfiles/helptext
   helpsplines bspline 7 ../../files/curves/help_B
   practicetrials 4
      practicesplines bspline 7 ../../files/curves/practice_C_01
      practicesplines bspline 7 ../../files/curves/practice_D_01
      practicesplines bspline 7 ../../files/curves/practice_A_01
      practicesplines bspline 7 ../../files/curves/practice_B_01
   experimenttrials 20
      trialsplines bspline 7 ../../files/curves/trial_D_05
         outdata Time_for_Trial
         outdata Control_Rating
         outdata Target_Rating
         outdata Subjective_Rating
      trialsplines bspline 7 ../../files/curves/trial_D_06
                       .
                       .
                       .
```

Figure 2.3: Example Session File.

them.

Experiment trial information is next, beginning at line 13 in Figure 2.3 (see Appendix C for a more complete example of this file). This first line of this part specifies the number of trials in the session; in this case 20. Then for each trial the controlled spline type and target curve are specified. The `outdata` lines provide space for the expected trial results produced by `CurveMatch`.

As the subject completes a session the corresponding session information is appended to the `outdata` lines, and written to the results files in the `files/results` directory. The input information is erased. If after a session the input file is empty then the subject is informed that the experiment has been concluded. Otherwise the subject is politely reminded to return for the next session.

2.3 The Trial Driver: CurveMatch

The most important part of the shape matching system, `CurveMatch`, is designed to execute a single trial. What follows is primarily a discussion of `CurveMatch`'s data flow, summarized in Figure 2.1, and a description of `CurveMatch`'s important routines.

`CurveMatch` input is comprised of two parameter files, created by `Control`, and a target curve definition file. The parameter files, `sessionrc` and `trialrc`, are kept in the `files/workfiles` directory and the target curve files are kept in the `files/curves` directory.

The `sessionrc` parameter file specifies the session, window and viewport information that remains unchanged from one match to the next. The `trialrc` parameter file specifies the controlled and target spline information and whether the trial is an experimental trial or a practice trial (see Appendix C for example parameter files). The controlled spline information consists of the spline type to be used and the number of control vertices. The target spline information consists of the file name containing a definition of the target spline.

The target spline information file contains the spline type, the number of control vertices for the control polygon and the x-y locations for each control vertex. All vertex data is in floating point ascii relative to a 0 to 1 viewport. By convention the data is stored in B-spline format and converted to the proper spline type when it is read into the program (see Appendix C for two examples of input target curves).

`CurveMatch` outputs two distinct streams of information: match analysis information and playback information. The analysis information is further subdivided into timing information and subjective and objective trial evaluation information.

Timing for the trial is in units of ten thousandths of a second. The single routine controlling the timer starts the timer at zero when first called during the trial. For every successive call it returns the elapsed time. The clock counter is limited to two and a half hours. Increasing this limit will require decreasing the timing granularity.

The *subjective evaluation* is the subject's self rating, obtained directly from the rating scale. The *objective evaluation* counts coloured pixels from the Iris frame buffer to calculate two different percentages. The first is the percentage of the target curve exactly matched by the controlled curve pixels. The second is the percentage of controlled curve pixels that are one pixel away from or on the target curve. The percentages are calculated at the end of the match by reading in the pixels from the Iris frame buffer and counting the coloured pixels. Again, the representative pixel colours are: green to indicate an exact match, blue-grey to indicate a close match, brown to indicate unmatched portions of the target curve and red to indicate unmatched portions of the controlled curve.

The playback information consists of time-stamped Iris input queue events. The stream of cursor moves and mouse click events is written into a file, which is specified by the `CurveMatch` command line "-o" option (`CurveMatch` is the only program that currently uses command line options). This playback file can later be read in by `CurveMatch` using the command line "-p" option in place of the "-o" option. This allows the subject's curve manipulation actions to be repeated and displayed in real time.

The most important routines in `CurveMatch` are those intimately connected with performing a match.

`CmInteract()`: This routine handles the subject's actions from start button selection through to rating scale selection. The start button selection initiates a call to `CmCurveManip()` and control returns when the stop button is selected, at which point the rating scale routine `CmGetRating()` is called.

`CmCurveManip()`: This routine takes care of all spline curve manipulations. It monitors the Iris queue and calls the proper routine, depending on the subject's action. The main routines called by `CmCurveManip()` are `CmShapeCurve()`, `CmMoveCurve()` and `CmTension()`.

`CmShapeCurve()`: This routine is responsible for changing the position of a control vertex and redisplaying the modified controlled spline.

`CmMoveCurve()`: This routine is responsible for handling translation requests.

`CmTension()`: This routine changes the tension of a single control vertex and redisplays the modified controlled spline.

`CmGetRating()`: This routine waits for the subject to select the rating scale boxes and returns the selected box number.

`CmQRead()`: This routine is called by all interaction routines. It is a cover routine for the Iris queue read function `qread()`. `CmQRead()`'s main purpose is to save the Iris queue events to file with a time stamp. If in playback mode ("-p" option) the queue events are read from the option specified file instead of the Iris event queue. It is the only routine which is aware of the difference between playback and normal trial mode.

An important element of CurveMatch is the implementation and use of splines. This is discussed in the next chapter.

Chapter 3

Computer Splines

Routines are available in the Iris system library for rendering splines. The chapter introductions purpose is to provide information for a more informed use of these Iris spline rendering routines [SGI 88]. The remaining two sections of this chapter deal with other aspects of splines in the tension experiment. Namely, the spline data structure, and the details of the tension interaction technique.

The splines of interest here are defined as piecewise parametric cubic polynomials. They are assumed by the Iris to have four components, x, y, z and ω per segment. Each component is described by a cubic polynomial and a parameter u, where u ranges between the values 0 and 1. Thus a spline segment can be written:

- $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$
- $y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$
- $z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$
- $\omega(u) = a_\omega u^3 + b_\omega u^2 + c_\omega u + d_\omega$

where the ω component is usually the constant 1 to provide correct mapping from four dimensional space to three dimensional space in which the Iris draws. For non-rational splines, ω does not vary with the parameter u. (The fourth homogeneous component, omega, ω is not to be confused with the italized w used for rational B-spline tension component).

The segment can also be written as the product of two matrices, where $S(u)$ is the resulting spline segment and N is the coefficient matrix.

$$S(u) = UN = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z & a_\omega \\ b_x & b_y & b_z & b_\omega \\ c_x & c_y & c_z & c_\omega \\ d_x & d_y & d_z & d_\omega \end{bmatrix}$$

The coefficient matrix N can be represented as the product of a control vertex matrix V and a basis matrix B. $S(u)$ can thus be written as the matrix product UBV; using a B-spline basis for B, the product $S(u)$ is:

$$S(u) = UBV = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1/6 & 3/6 & -3/6 & 1/6 \\ 3/6 & -6/6 & 3/6 & 0 \\ -3/6 & 0 & 3/6 & 0 \\ 1/6 & 4/6 & 1/6 & 0 \end{bmatrix} \begin{bmatrix} e_x & e_y & e_z & e_\omega \\ f_x & f_y & f_z & f_\omega \\ g_x & g_y & g_z & g_\omega \\ h_x & h_y & h_z & h_\omega \end{bmatrix}$$

Successive segments $S(u)$ can be formed in the usual way by using overlapping control vertices [Bartels 87].

At null tension ($w = 0$, $\beta_2 = 1$), the rational B-splines and the Beta2 splines are simply uniform cubic B-splines. Rational B-spline tension is achieved by letting ω vary with the parameter u and thus the x, y, z polynomials are all divided by a second cubic polynomial. The ω value is multiplied though the other coordinates in order to satisfy the Iris mapping from four dimensional computation space to three dimensional rendering space. Thus the component matrix V becomes:

$$V = \begin{bmatrix} e_x e_\omega & e_y e_\omega & e_z e_\omega & e_\omega \\ f_x f_\omega & f_y f_\omega & f_z f_\omega & f_\omega \\ g_x g_\omega & g_y g_\omega & g_z g_\omega & g_\omega \\ h_x h_\omega & h_y h_\omega & h_z h_\omega & h_\omega \end{bmatrix}$$

Beta2 spline tension is somewhat more complex. Each segment polynomial is a function of four β_2 values. When one of these changes value, the segment polynomials, which is to say the basis coefficient matrix, must be re-computed before the segment can be redrawn [Bartels 84]. Due to the speed of the Iris and the small number of segments being displayed the complete curve is recomputed and redrawn rather than recomputing only the affected segments.

The Iris draws a segment as a series of straight lines using a method called *forward differencing* [SGI 88, Bartels 87]. The programmer must supply a curve precision n which specifies the number of straight lines used to draw one segment. A forward difference matrix M is formed from a basis matrix B, a control vertex matrix V and a curve precision matrix F as follows:

$$M = FBV = \begin{bmatrix} -6/n^3 & 0 & 0 & 0 \\ 6/n^3 & 2/n^2 & 0 & 0 \\ 1/n^3 & 1/n^2 & 1/n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} BV$$

The last row of M is a point on the curve while the other rows represent differences for steps along the curve. Subsequent points are formed by summing adjacent difference rows with the matrix multiplication:

$$M^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} M$$

The software algorithm that mimics forward differencing algorithm as implemented in the Iris hardware is:

```
move (M[3][0]/M[3][3], M[3][1]/M[3][3], M[3][2]/M[3][3]);
for ( count = 0; count < n; count++ )
{
  for ( i = 3; i < 0; i-- )
    for ( j = 0; j < 4; j++ )
     M[i][j] = M[i][j] + M[i-1][j];
  draw (M[3][0]/M[3][3], M[3][1]/M[3][3], M[3][2]/M[3][3]);
}
```

This has been implemented in software to verify that the curves the Iris claims to produce can also be produced, albeit a bit slower, with a non-Iris dependent method.

The next two chapter sections relate these splines back to the experiment software system. The first section details the object-oriented spline data structure model. The second presents an overview of the design implementation and functions related to the incorporation of tension.

3.1 Spline Objects

Within the experiment system a robust spline data structure has been implemented using C data structures containing generic function pointers[1]. Spline type specific in-

[1]C++ was considered, but was not available on the Iris for the coding stage.

formation is encapsulated by having each spline supply its own set of functions whose parameter lists match those of the generic function pointers. These basic building blocks are displayed in Figure 3.1. Each function implementation is dependent on

```
typedef struct SplineProcs
{
    char  *(*name)();
    int    (*alloc)();
    int    (*dealloc)();
    int    (*changebasis)();
    Coord *(*getsegment)();
    int    (*render)();
    Coord *(*defbasis)();
 } SplineProcs;
```

Figure 3.1: Spline Function Pointers.

details of the spline type being represented. These generic functions are:

`name()`: returns the name of the spline type. Each spline has a unique descriptive name that identifies the spline in a human-readable form (ASCII). It is usually used for input and output of splines curves.

`alloc()`: reserves the exact space required by the spline. A minimum for every spline would be to allocate space in which to store the control vertices.

`dealloc()`: deallocates the memory reserved by alloc().

`changebasis()`: because all splines are stored in B-spline format, other spline types must transform the input control vertices from the B-spline basis to their own.

`getsegment()`: returns the control vertices for a spline segment. It is an iterator as it will return successive segments on each call and return zero if there are no segments remaining. It can be reset at any time to return the first segment once again. This ability is required to perform analysis on a specific segment of the spline, or for drawing the spline independently of the Iris (this code is available for future ports).

`render()`: draws the spline. It is currently dependent on the Iris drawing routines.

`defbasis()`: sets up the proper basis matrix in the Iris hardware. It also returns a pointer to this basis for any routine that needs the information. A pointer to the basis matrix is needed for code that draws the spline independently of the Iris.

The information for a spline is not only composed of function pointers but several other attributes. The complete spline definition is given in Figure 3.2. A description of each attribute is:

```
typedef struct Spline
{
    int            id;
    int            numcontrols;
    Boolean        controlson;
    short          colour;
    int            nc;
    Coord          *curve;
    SplineProcs    *procs;
    char           *info;
} Spline;
```

Figure 3.2: Spline Type Definition.

`id`: This variable provides a unique numeric identification for the spline basis. It also distinguishes between tension splines (negative value) and non tension splines (positive value).

`numcontrols`: This field specifies the number of control vertices.

`controlson`: This flag dictates whether or not the spline's control polygon is visible, i.e.: on or off.

`colour`: This is the spline's colour index into the Iris colour map.

`nc`: This attribute represents the number of relevant components of the coordinate matrix. This is commonly used when indexing the curve from vertex to vertex. Some routines need to know the number of vertex components to skip to access the next vertex. For example there may be three components x, y and z or four components x, y, z and tension.

`curve`: This one dimensional array of floating point values contains the location of a curve's control vertices. The routines that index the vertex values are considered to have intimate knowledge of the spline and should belong to the `SplineProc` structure or be grouped with other similar spline code.

`procs`: This field is a pointer to spline function pointers as detailed previously in Figure 3.1.

`info`: This extra space is provided for individual spline use, if ever required. It must be cast into the desired data type.

This organization facilitated a clean implementation and should make it easy to add other spline types. For example the incorporation of the two tension spline types was straightforward.

3.2 Tension

Developing an adequate interface for adjusting tension absorbed a substantial part of the software development effort, partly because determining the value of tension was the primary objective, and partly because the task presented some intrinsic difficulties.

Design of the tension interface was crucial since an inappropriate interface might bias the experiment against tension. Thus the interface was required to have:

- a simple and intuitive method for adjusting tension,
- a smooth mapping between hand motion and curve change,
- the ability to recognize negative, positive and starting (*null*) tension,
- the ability to easily differentiate between tension and non-tension splines,
- all changes in tension reflected in the spline curve so as to provide visual feedback.

Many interaction techniques were considered, and a number implemented for evaluation purposes. The principle alternatives – digit control, light handle, and stirrer – are sketched and discussed below.

The first method considered was a tension control window containing digits and buttons for tension feedback and adjustment. Thus the tension could be changed slowly when modifying the 1's digit and changed quickly when modifying the 1000's digit. This method was perceived as too complex and time consuming. Additionally, it would distract the subject from the spline.

A second possibility was a mouse driven light handle [Newman 79]. The y component could provide the tension change while the x component provided the magnitude of the change. Some sort of numerical feedback would also be necessary. The main perceived problems were that the interaction model and numerical feedback would not be adequately intuitive for naive users. An additional problem is that the technique would remove the subject's attention from the spline.

A third possibility was to use a stirrer [Evans 81]. In this technique, circular cursor motions would be used to increase or decrease tension, depending on the motion direction. A prototype stirrer was implemented using the original approach developed by Evans *et. al.*:

1. sample the last three cursor locations,
2. form two vectors using the first and middle location and the middle and last location,
3. calculate the angle of rotation by the angle between the two vectors,
4. determine the direction of movement using the normal to the two vectors.

An additional option would be to use the angle magnitude to scale the amount of tension change.

The stirrer method was partially successful, but failed in many respects. Any deviation from a circular motion caused unsmooth changes in the controlled spline. Moreover, the technique did not reflect the tension's relation to the control vertex basis since the stirring motion could be performed anywhere on the screen. Eventually the stirrer method of calculating the tension change was rejected, although the concept of circular motion was retained.

In the technique finally adopted, the circular motion is anchored at the selected control vertex. The tension is changed by moving the cursor clockwise about the control vertex to increase the tension and counter-clockwise to decrease the tension. The tension is a function of the total angle that the subject has traveled around the control vertex. Positive angles are mapped to positive tension, while negative angles are mapped to negative tension. A finer degree of control is achieved by moving the mouse away from the control vertex. This provides a smaller angular change for the same amount of motion when closer to the control vertex. Overall, this technique mimics the experience of tightening or loosening a jar lid or screw. Similarly it can be viewed as turning a winch handle where the curve is "winched" towards the control vertex.

When a tension control vertex is selected the control polygon disappears and a larger tension-feedback circle surrounding the control vertex is displayed (see Figure 3.3). This larger circle is originally the same grey shade as the control vertex, indicating the null tension point. As positive tension is applied, a bright arc whose extent corresponds to the amount of applied tension overlays the circle. Negative tension is similarly handled but represented by a dull arc. The arcs grow from 0 degrees to 360 degrees for positive tension and 0 to -360 degrees for negative tension. Thus full circles denote the maximum or minimum range of tension as allowed in the experiment (this tension clamping will be explained later). Thus by selecting a control vertex, the subject can tell at a glance whether the tension is positive or negative, and roughly how much tension has been applied. Through personal experimentation, a one to one proportion was deemed most suitable when mapping the subject's angular movement to the change in the feedback arc. Thus a one degree arc movement by the user increases the tension arc by one degree

The function relating the amount of tension to the total angle traveled by the

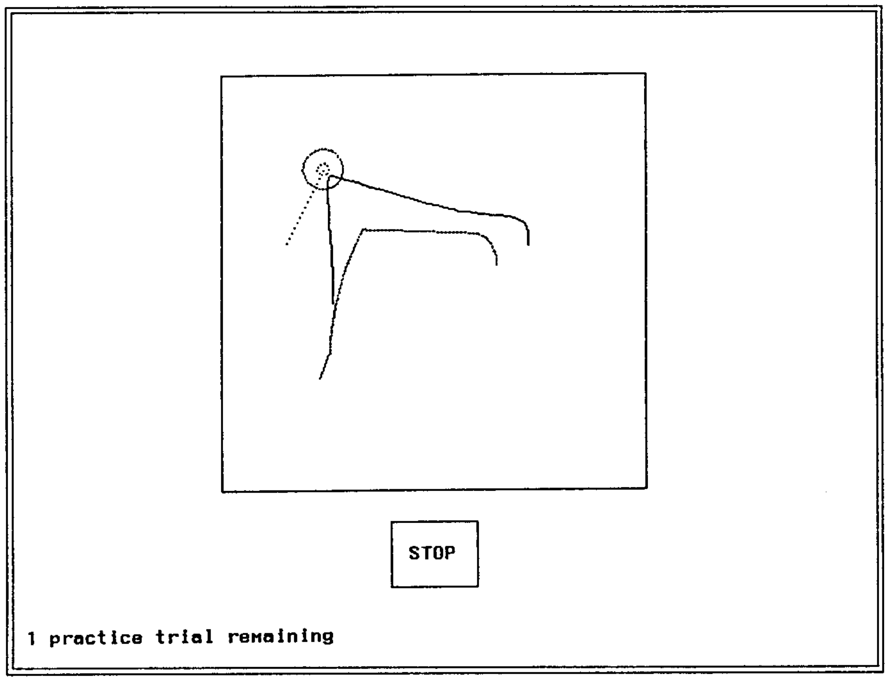


Figure 3.3: Tension User Interaction.

subject is

$$tension\_amount = (total\_angle/scale)^3$$

or conversely

$$total\_angle = scale\sqrt[3]{tension\_amount}$$

This function was chosen for many reasons: simplicity, sign preservation while mapping from angles to tension, and space efficiency (since only the tension value is saved). Finally, it also provides an approximately constant rate of change in the shape of the spline as a function of angular displacement. Curves reacts quickly for changes at lower tension values and slowly for changes at higher tension values. This function counterbalances this effect by providing rapid change for high tension values and slow change for low tension values. Moreover the curve responds less rapidly to changes in tension as the angle crosses the zero degree location, making it easy to reset the tension to its initial null value.

The code implementing this interaction technique exists in a single file called `CmTension`. The only other code elements concerned with tension are the individual tension spline routines. These routines need only supply a minimum tension, a null tension and a maximum tension.

Decreasing and increasing tension are potentially infinite processes. What will follow is a discussion of how these two infinite elements where made manageable for the tension implementation. The main problem is how to give the user feedback. For example representing tension from its practical range to the machine limit is not possible with a feedback circle allowing only a single revolution. Different colours could be used for different revolutions, but this would distract from the curves. Other solutions were tried but the most obvious and easiest was to limit, clamp, tension within practical limits. How these limits were found is detailed below. As a visual supplement to Figure 3.3 to the discussion a Beta2 spline with full tension on two internal control vertices is shown in Figure 3.4.

Decreasing tension pushes the spline away from the associated control vertex. With negative tension there is an extra problem having to do with division by zero. For example, within the rational B-spline this value is zero since the tension value is used as the divisor for the x, y and z coordinates. Likewise the Beta2 spline calculations will cause division by 0 if all four Beta2 values for a segment are -4 simultaneously. Clamping the negative tension ($w = 0.01$, $\beta_2 = -3.99$) avoided extra code that would check for division by zero. Of course these limits can be avoided but negative tension, practically speaking, was not expected to be of much use during the experiment since unclamped negative tension allows the creation of non-intuitive spline curve shape and behaviour. The permitted negative tension was sufficient to create curves that covered the convex hull.

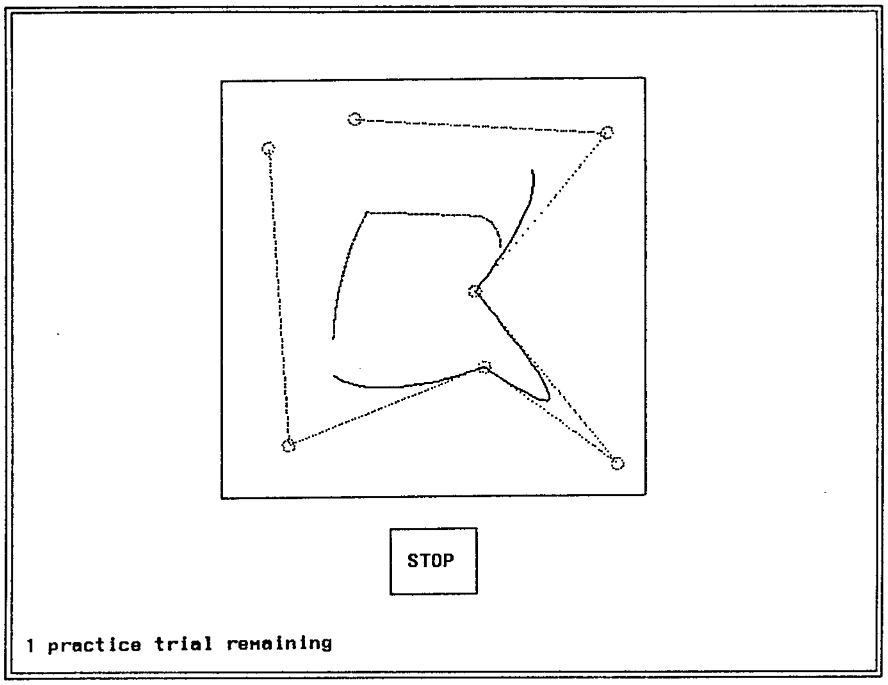


Figure 3.4: Positive Beta2 Tension.

Increasing tension pulls the curve towards the control vertex associated with that particular tension parameter. A clamp was put on positive tension because of the spline reaching a point which is for all practicality at infinity. This point is loosely defined as when the spline curve passes through the center of the control vertex.

It does so because the basis function associated with the control vertex approaches 1 at its midpoint as tension approaches infinity. The maximum allowed values for tension ($w = 2001.0$, $\beta_2 = 2000.0$) were chosen by working with the basis function for the associated control vertex. The percentage of the basis function at its maximum value, K, is considered as it is affected by tension associated at that control vertex with all other control vertices at null tension. This was decided upon to provide a clean precise metric for deciding a clamp on the possibly infinite positive tension.

The basis function percentage chosen was $K >= 99.9$. This gives the "practical infinity" plus some margin. The values for w and β_2 given above are obtained by symbolically solving the basis polynomial $s_1(u)$ at $u = 0$ for the tension values that yield the desired K which lies between 0 and 1 (see Figure 1.3).

For Beta2 splines substitution of 0 into the equation of the segment polynomial $s_1(u)$ (see formulas for $s_1(u)$ on page 8) yields:

$$s_1(0) = (\beta_2 + 4)/(\beta_2 + 6) = K$$

This formula is rearranged to obtain the maximum tension value to be allowed.

$$\beta_2 = \frac{6K - 4}{1 - K}$$

The same method is used for rational B-splines. The rational basis function, $B_i^*(u)$ is given by:

$$B_i^*(u) = \frac{w_i B_i(u)}{\sum_{j=i-3}^{i} w_j B_j(u)}$$

Since all of the basis functions sum to one:

$$B_i^*(0) = \frac{w_i B_i(0)}{w_i B_i(0) - [1 - B_i(0)]}$$

Then solving for $s_1(0)$ gives:

$$s_1^*(0) = \frac{w_i s_1(0)}{w_i s_1(0) - [1 - s_1(0)]} = K$$

and is rearranged to obtain the maximum tension value for the desired percentage.

$$w = \frac{K}{2(1 - K)}$$

This concludes the discussion of the tension aspect of splines. The next chapter presents the target spline generating process.

Chapter 4

Target Curves

For the tension experiment, target curves are derived from real objects. The change from random target curve generation has many advantages, but also a few disadvantages.

Relative to Bosch's experiments, the question of whether a target curve is mathematically representable as a particular spline is moot. Instead the question is whether a given spline can be reshaped so as to pass through a set of sampled pixels and only those pixels. It may, of course, be the case that a given target curve cannot be exactly matched but the extent to which that is true for different spline classes is itself relevant because the target curves have been chosen for their intrinsic interest (target curves from letters are always of interest to the font industry). Unfortunately generating and verifying a set of real life curves is much more time consuming than randomly generating spline curves.

This chapter is concerned with the derivation of target curves from a set of foot-high master fonts. In the first section the font itself is described. In section 4.2: Processing, the curve generating procedure is discussed.

4.1 Letters

The model letters are taken from a font set meant for teachers of the art of graphic font design [Price 61][1]. The letters are a classic Roman font. Classic Roman letters have been studied by many people, including Leonardo da-Vinci, who desired to construct a set of letters that best reflect the idealized Roman form. The classical Roman letter is important because it is "stressed as the standard of all Roman capital letters, the point of departure from which all developments, including the most modern Twentieth century forms of today have come." As such it is a fitting letter with which to start real life curve experiments.

From the font designer's point of view the Roman letters can be divided into four design groups. These are:

1. straight lines and right angles consisting of the letters: E F H I L T
2. straight lines and oblique angles consisting of: A K M N V W X Y Z
3. straight lines and curves consisting of: B D J P R U and
4. curvilinears: C G O Q S.

The target curves were generated only from the third and fourth design groups as those were the only two groups containing curves.

For this experiment curve selection was limited to the upper case Roman letters. There should be ample curves for future experiments from other sets published by Price. These consist of the "Roman Lower Case Letter; Italic and Script Forms; Medieval ('Old English'); Numerals and Nineteenth Century Styles". There is also

[1]All quotes and information in this section was gathered from this syllabus.

no reason to remain strictly with English fonts as the Hebrew, Chinese and Arabic forms, among many, will provide an almost infinite variety of curves.

It is interesting to note that the authors say that "the essential of good lettering is human skill, tempered with patience and a desire for fine results." Again and again the authors stress not the mechanics of letter design but the artist's feel for the form of the letter and attention to detail. It is sincerely hoped that the same sentiments are carried over to the use of splines when matching real life objects.

4.2 Processing

The basic curve processing is to digitize portions of the letters and then use various software tools to find the letter edges. The final curve representation is a spline whose control vertices are the pixel locations of the digitized curve. This representation was chosen for continuity with the current target curve representation method used in the shape matching system and to avoid the slowness of dealing with large bitmaps.

The curve processing hardware consists of a sony colour video camera, a light table, an Ikonas frame buffer, a Mac II and a Vax 8600. The software consists of the Ikonas control software tools and the Im tookit developed by Alan Paeth [Paeth 86]. These software and hardware tools were readily available in the Computer Graphics laboratory and were chosen as the first available means of generating the desired target curves.

The first step is setting up the equipment. This should only be done once to ensure a consistent picture taking environment. For this the camera is set up on the light-stand with the letter placed beneath. The desired letter portion is framed with pieces of blank white paper and secured with a glass cover. The lights in the room, except for the light table, are turned off to avoid reflection on the glass.

The following steps are repeated once for every desired letter portion. The first step is taking the picture. This involves clearing the Ikonas to its initial state with the command

```
iknuke
```

Then a still photo is captured from the sony camera into the Ikonas buffer using

```
ikgrab
```

which stores a digitized frame in the Ikonas frame buffer. The solid letter portion achieved is a filled in version of the letter outline as shown in Figure 4.1.

The image is then copied from the Ikonas to the Vax 8600 in Im format by

```
imikr > curve1
```

where `curve1` is a temporary processing file.

From this point on the only software setup task is the creation of the kernal matrix required by the edge detection tool. The kernal matrix has the values:

$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

These values are stored in a file to be accessed during the generation process.

At this stage the letter is a 24 bit RGB image. The image is subsequently converted to an 8 bit greyscale image by extracting the red component using:

```
imexpr curve1 -p n8 -e r > curve2
```

The image in the second temporary file `curve2` is then converted to a binary image by thresholding; that is, all pixels above half intensity are mapped to white and all pixels below half intensity are mapped to black. The thresholding command is:

```
imexpr curve2 -p n1 -e n > curve1
```

At this point the edges are extracted using

```
imkern curve1 kernal -offset 1 > curve2
```

where `kernal` is the previously mentioned kernal matrix. This command detects the borders of letters by scanning the image and combining adjacent pixels according to the given kernal matrix. The kernal matrix used turns a pixel to black if it is the same as its neighbours. Thus imkern detects the borders of letters where there is a sudden change from black to white. This is an excellent, elegant solution as the edge remaining after imkern is one pixel wide and requires no line thinning algorithm[2].

Because of imprecise mapping of the camera image to the Ikonas frame buffer there is an edge detected along the image frame. This is cropped away using:

```
imcrop curve2 -x 30 -y 30 -h 450 -w 450 > imletter
```

At this point a fairly clean binary image of the letter edges exists in the file imletter. An example is shown in Figure 4.1.

The next step is bit-editing the letter edge image. This is necessary to retrieve the edge portion that will eventually become a target curve. Since there was no Im bit editing tool an adhoc method was devised. The Im image is transformed to MacPaint

[2]Edge detection is rarely this easy. It helps that the initial images are this simplistic.

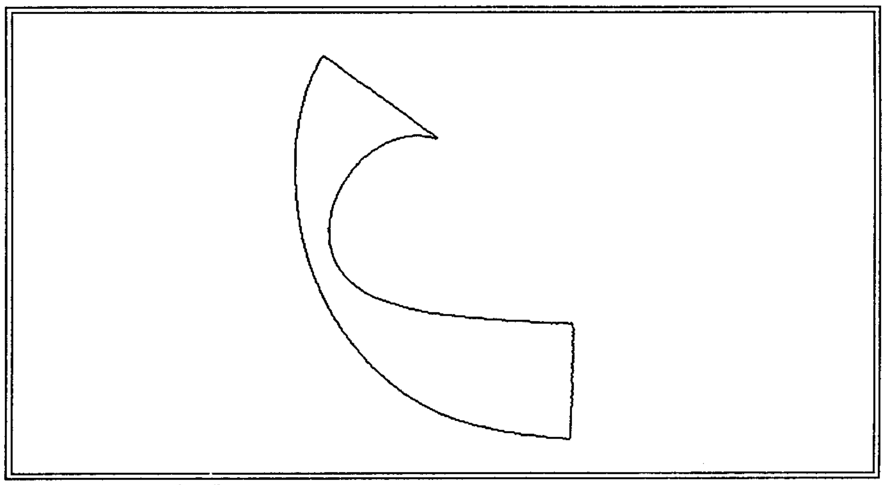


Figure 4.1: Letter Edges.

format, edited using SuperPaint on a Mac II and retransformed into Im format. The transformation to and from MacPaint is achieved using tools written by Bill Wallace, a CGL co-op student. Also during this stage there is a small amount of pixel repair. This necessity is obvious by looking closely at the top straight line portion of the letter outline provided in Figure 4.1

The last task is transforming the Im image to spline format without changing the appearance of the curve. To do this it is necessary to preserve the pixel locations of the original Im curve, by making the Im image viewport the same size as the experiment viewport. In actuality this is not difficult as the Im viewport always had to be enlarged.

Now that the pixel locations are guaranteed to be identical the pixels need to be represented in a form that can be easily drawn and translated. The easiest form to use with the shape matching system is to make each pixel a control vertex and pass a spline through each pixel. The command that performs the transformation of an Im file to the experiment spline format is (an Im tool written by the author):

```
cat Im_curve | ImMatch target_curve
```

Since the B-spline used to draw the target curve is not an interpolating spline, the resulting screen pixel locations were checked against the actual pixel locations to ensure identical curves.

With a process for obtaining target curves the only remaining chore is to run and evaluate the tension experiment.

Chapter 5

Tension Experiment

The weights in a rational B-spline and the β_2 parameters in a Beta2 spline provide a tension-like facility: by manipulating them one is able to draw the curve up against a portion of the control polygon, and a sufficiently, large individual weight or β_2 value results asymtoticaly in the interpolation of a control vertex. Owing to the convex hull property, the latter generally results in the appearance of a sharp corner or *cusp*.

The experiment goal was to gain insight into the practical usefulness of these tension parameters in interactive curve design. More precisely, there were three interesting questions to be looked at:

1. Does the addition of tension make curve design easier in comparison to a non tension curve with the same number of control vertices?
2. If so, is the manipulation of tension parameters of greater or lesser value than the manipulation of a non tension curve with a larger number of control vertices?
3. Is either form of tension easier to use than the other?

To explore these questions subjects were asked to match equally represented cusp and non-cusp curves with one of the following four kinds of splines (*controlled splines*):

- Beta2 spline with seven control vertices (Beta).
- Rational B-spline with seven control vertices (Ratb).
- B-spline with seven control vertices (Bsp7).
- B-spline with ten control vertices (Bs10).

The inclusion of ten vertex B-spline curves was motivated by the second question; the choice of these additional control vertices was to some extent arbitrary, given the following considerations:

1. Potentially likely places to use tension were at the two endpoints and at interior cusps, of which the target curves had at most one.
2. Exactly overlaying three consecutive control vertices also provides a cusp, though one with short straight line segments close to the cusp. Overlaying two control vertices causes a sharp bend in the curve but generally without interpolation of the control vertex.
3. Providing as many degrees of freedom to a B-spline curve as with a rational B-spline curve or Beta2 spline curve would have required using a 14 vertex B-spline. However it did not seem likely that subjects would actually use all seven tension parameters and it did seem possible that the effort involved in manipulating all 14 B-spline control vertices might mask improved performance obtained from the availability of a small number of additional control vertices.

4. Too much time would have been required to study B-spline curves having several different numbers of control vertices.

The remaining sections in this chapter describe the following experimental elements:

1. The experiment subjects: the details of the subjects' background knowledge and the criteria considered when selecting them are presented.
2. The experiment procedure: this encompasses everything from the subjects' selected to what the subject sees and experiences during the experiment. These experiences include the room, the equipment location, the lighting, the comment forms and the general experiment flow.
3. The methodology used: that is the blocking of the target and controlled spline types.
4. The results: a straightforward analysis of the experiment results and a summary of the subjects' written comments are presented.
5. A discussion of the results.

5.1 Procedure

The subjects and the procedure they follow from the moment of contact through experiment completion is detailed in this section. For the tension experiment there were twenty-four paid subjects that took part. The subjects were recruited from the University of Waterloo psychology pool and through the local University electronic news system.

A summary of the subjects is listed in Table 5.1. Note that an x beside the mouse experience field indicates that the subject used a mouse on more than one computer and is thus rated as an experienced user. To qualify, a subject is

Subject	Age	Sex	Previous Experience: Spline/Mouse	Subject	Age	Sex	Previous Experience: Spline/Mouse
1	20	M	no / yes	13	23	M	no / yes
2	21	M	no / yes x	14	23	F	no / yes
3	34	F	no / yes	15	19	M	no / yes
4	30	F	yes / yes	16	22	F	no / yes
5	19	M	no / yes	17	23	F	no / yes
6	32	M	yes / yes x	18	24	M	no / yes
7	27	M	no / yes	19	22	F	no / no
8	24	M	yes / yes	20	21	F	no / yes x
9	19	F	no / no	21	22	M	no / yes
10	23	F	no / yes	22	23	M	yes / yes x
11	21	M	no / no	23	22	F	no / yes
12	21	M	no / yes x	24	24	M	yes / yes x

Table 5.1: Experiment Subjects. X's indicate experienced users.

required to have no uncorrected visual problems and a good comprehension of written English. Additionally, the subject is not allowed to have worked with computer generated splines but is allowed to have a slight knowledge of spline mathematics. Spline familiarity was commonly gained from the courses CS377 (Introduction to Numerical Analysis) and, in one case, CS488 (Introduction to Computer Graphics). In CS377 subjects are not introduced to non-parametric splines while in CS488 Bézier surfaces are introduced for the purpose of an assignment.

Initial contact with the subject is either by phone or electronic mail. The subject is told the name of the experiment, the payment to be received for completing the experiment (40 dollars), the approximate duration of the experiment (five hours) and its organization (four distinct sessions). If the subject is interested, a time is scheduled and the experiment room location and phone number is given.

When the subject arrives for the first session, a consent form and an information form are filled out. The consent form ensures that the subject understands that the experiment is performed under the guidelines of the Office of Human Research at the University of Waterloo. The information form is needed to obtain pertinent facts, including those summarized in Table 5.1.

After these preliminaries, the subject is seated in an "office-size" room containing a Personal Iris workstation, a 19 inch colour display monitor, a keyboard and an optical three button mouse. Ambient illumination is provided by a single dim light behind the monitor, facing up and away from the subject. This effectively eliminates screen glare. The monitor is set to its maximum contrast and the intensity is adjusted downward from the maximum to produce a black background.

The experimenter then logs into the shape matching account and leaves after indicating that all instructions will be provided by the computer. At this point, the subject enters an 8 digit experiment identification number to start their individual session. Before the experimental matches of the first session, the subject reads the `Help` instruction pages. At this stage the four practice trials and the 20 experimental trials for the session are presented for the subject to match. The remaining three sessions are identical, using different controlled and target curves. However, the subject has a choice of reviewing the instruction pages.

During a single trial the target curve and controlled curve are displayed in a centralized match window. The subject manipulates the control vertices and tension parameters to cause the controlled curve to overlap the target curve end to end. Good matches are keyed by a change in the colour of the controlled curve. When a portion of the controlled curve is one pixel away, that red portion changes colour to a bluish grey. When a portion of the controlled curve overlaps the brown target curve the overlapping portion becomes green. What the subject attempts to do is to minimize the amount of red and maximize the amount of green. When the match is satisfactory and performed within a reasonable amount of time (at the subjects discretion) the subject stops the match and rates the quality of the match on a scale of one (poor) to seven (very good).

After completing all the trials in each session, the subject completes a session-dependent comment form. After the first session the subject answers the question:

- "Did you have any difficulty understanding the instructions?"

After every session, including the first, the subject replies to the questions:

- "Did you use a specific strategy in moving controls to achieve a match?"
- "What criteria did you use in rating the quality of your match?"

After the second, third, and fourth sessions, the subject answers the questions:

- "In which session did you find the control of the curves easiest?"
- "In which session did you think that you were most successful in your final matches?"
- "Which session did you enjoy the most?"

Subjects are permitted to schedule subsequent sessions at their convenience. At the end of the last session the subject is paid and signs a form indicating that payment has been received.

5.2 Blocking

Blocking, as the term is used here, is the careful arrangement of which spline type and spline set is presented to a subject at a specific time. The purpose of blocking is to avoid bias within the experiment (see [Ostle 75] for a more detailed description of experiment setup and description of blocking). This section shows how the permutation of the spline types and the permutations of the spline sets are combined evenly across subjects and sessions so as to avoid bias.

The target curve sets are assigned meaning and grouped as follows:

- A B: are two sets of curves without cusps.
- C D: are two sets of curves with cusps.

Every session uses curves from one non-cusp and one cusp curve set. This results in the four possible combinations AC, AD, BC and BD, which means that there are 4! = 24 unique cusp/non-cusp pair orderings. Since each of the 24 subjects matches one of the possible cusp/non-cusp pairs in each session and since each target curve set (e.g. A) is comprised of 10 curves, the subject will see each of the 40 individual target curves twice.

Likewise the four controlled spline types have 24 possible unique orderings and Table 5.2 displays the result of combining the permuted curve sets and the permuted spline types. For example, line 1 shows a first session with a seven control vertex B-spline controlled curve (Bsp7) using target curves from the non-cusp set A and the cusp set C.

For each session column, the combination of the permutations is arranged so that each cusp/non-cusp pair appears with the same spline type as evenly as possible.

Line	Session 1	Session 2	Session 3	Session 4
1	Bsp7-AC	Ratb-BD	Beta-AD	Bs10-BC
2	Bsp7-AC	Ratb-BD	Bs10-BC	Beta-AD
3	Ratb-BD	Bsp7-AC	Beta-AD	Bs10-BC
4	Ratb-AC	Bsp7-AD	Bs10-BD	Beta-BC
5	Bsp7-BD	Beta-AC	Ratb-BC	Bs10-AD
6	Bsp7-AD	Beta-AC	Bs10-BD	Ratb-BC
7	Beta-AC	Bsp7-AD	Ratb-BC	Bs10-BD
8	Beta-BD	Bsp7-BC	Bs10-AD	Ratb-AC
9	Bsp7-BC	Bs10-AC	Beta-BD	Ratb-AD
10	Bsp7-BC	Bs10-AC	Ratb-AD	Beta-BD
11	Bs10-AD	Bsp7-BD	Beta-BC	Ratb-AC
12	Bs10-BC	Bsp7-BD	Ratb-AD	Beta-AC
13	Ratb-AC	Beta-BC	Bsp7-BD	Bs10-AD
14	Ratb-AD	Beta-BD	Bs10-AC	Bsp7-BC
15	Beta-BC	Ratb-AD	Bsp7-BD	Bs10-AC
16	Beta-BD	Ratb-BC	Bs10-AC	Bsp7-AD
17	Ratb-BC	Bs10-BD	Beta-AC	Bsp7-AD
18	Ratb-BC	Bs10-AD	Bsp7-AC	Beta-BD
19	Bs10-AD	Ratb-AC	Beta-BC	Bsp7-BD
20	Bs10-BD	Ratb-AD	Bsp7-BC	Beta-AC
21	Beta-AD	Bs10-BC	Bsp7-AC	Ratb-BD
22	Beta-AD	Bs10-BC	Ratb-BD	Bsp7-AC
23	Bs10-AC	Beta-BC	Bsp7-AD	Ratb-BD
24	Bs10-BD	Beta-AD	Ratb-AC	Bsp7-BC

Table 5.2: Experiment Blocking.

Since each spline type appears in each column six times, two cusp/non-cusp pairs must be repeated in a column. Again referring to the table, the two cusp/non-cusp pairs not repeated in column one must be repeated in column two. For example, Bsp7 is coupled in column one with the cusp/non-cusp pairs (AC AC BD AD BC BC) and column two with the cusp/non-cusp pairs (AC AD AD BC BD BD). The same technique is used for the last two sessions as well.

The individual target curves within a session are randomly ordered. The only constraint is that the same number of cusp and non-cusp curves appear in each half of a session. This is accomplished by using the first five cusp and first five non-cusp curves that appear in a random ordering of the two curve sets. The purpose is to be able to study the last half of each session in case the learning process is too large a factor to be dealt with by the practice curves. The separate practice curves are also randomly ordered. One extra curve is supplied from each set, A, B, C and D as a practice curve. These practice curves are retained, and randomly ordered, for the start of every session.

The lines in Table 5.2 are randomly ordered so that it is undetermined as to which subject will receive a particular spline ordering. This is to ensure that the subjects coming later in the experiment will not be biased towards a certain spline type due to the structure of the ordering. For example, without a random ordering of the subject definition lines, the Bsp7 technique in session one would be manipulated solely by the first 12 people taking part in the experiment.

This blocking makes it possible to perform meaningful statistical analysis on the experimental results.

5.3 Results

The experiment produced a number of interesting and surprising results. This section first highlights general observations on the subject's spline manipulation techniques and then summarizes the actual experiment results.

From both personal hands on experience and viewing the experiment playback data, divergent tension traits were observed between the rational B-spline tension aspect w, and the Beta2 tension aspect β_2, and then summarized below:

Maximum tension: When all vertices are at maximum tension, the Beta2 spline is composed of straight line segments. Correspondingly, the rational B-spline acts as if no tension has been applied because the tension at each control vertex counterbalances the neighbouring vertices tension.

Stretching effect: The rational B-spline has a tendency to alter noticeably the length of the curve when the spline curve is pulled towards the control vertex. For instance, turning up the tension on two interior adjacent control vertices for a rational B-spline curve will cause the complete curve to appear as a single line between the two vertices. This is shown in Figure 5.1. This figure is the counterpart to a previous tension figure for Beta2 splines shown in Figure 3.4 on page 45. The same adjustment on a Beta2 spline results in the appearance of a straight line between the two control vertices but with the curve seeming to continue in a normal manner outside the control vertex interval.

Stiffness effect: Another property of the rational B-spline is that the curve appears to become inflexible near control vertices with high tension. When repositioning the neighbouring control vertices, the curve acts more like a straight line than a spline, making it difficult to shape the curve. This property does not manifest itself in the case of the Beta2 splines.

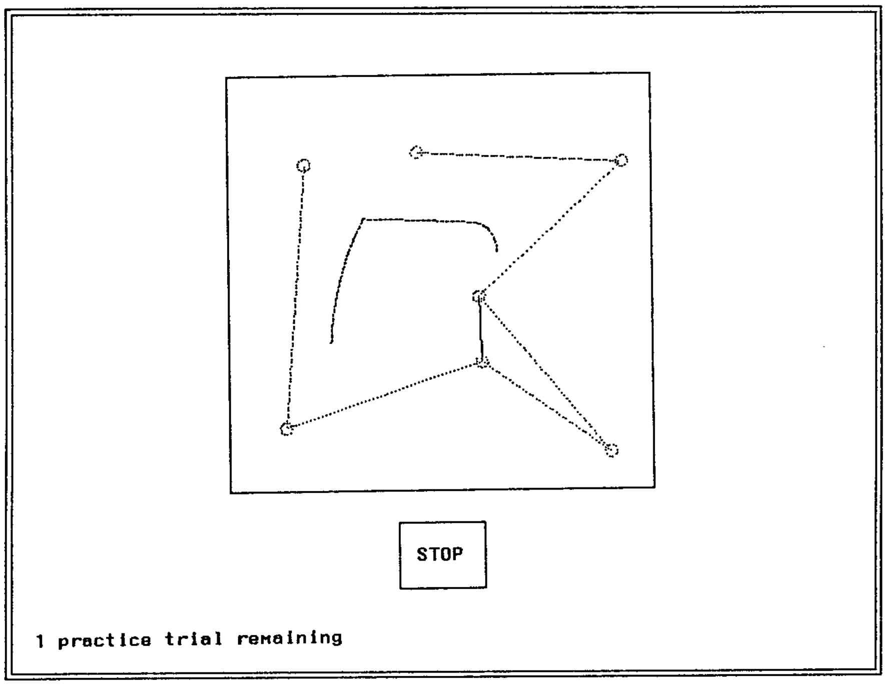


Figure 5.1: Positive Rational B-spline Tension.

The most surprising observation was that some subjects did not understand how to resize the controlled curve to match the size of the target curve. This took the form of a matching technique that was labeled *overlay match*, in which the subject would completely cover the brown target curve with a much longer red controlled curve. Thus excess red controlled curve portions would extend from the ends of the overlayed target curve.

Some subjects attempted to eliminate the excess portions by wrapping the controlled curve ends back onto the target curve. Others assumed that the match was sufficient and proceeded to the next trial. Interestingly enough, as the subjects progressed through the sessions, most apparently realized that they could resize a controlled curve and would begin *exact matching*; that is, not only overlaying the curves but also making both curves the same size. The experienced mouse users experienced no such confusion performing *exact matching.*

The *overlay match* was discovered by watching the playback data of the subjects. It was decided that the effect of tension would be best evaluated across a single match method, either overlay or exact, but not both simultaneously. To determine objectively which group of subjects to replace, it was required that the subjects obtained, on average, 80 percent of the target curve either close to the controlled curve (one pixel away) or exactly on the controlled curve. Since the subjects using the overlay method had consistently lower objective ratings in the first session they were eliminated. In all, ten replacement subjects were needed. Of the ten replaced subjects, eight used the overlay at one time or another in the experiment and two were novice users. The novice users had severe matching problems in the first session and ended with 60 percent ratings (although they tried very hard). The 10 replacement subjects were given a one sentence verbal emphasis to match the curve sizes as well as overlay the two curves.

5.3.1 The Main Analysis

The main objective and subjective experiment results are summarized in Table 5.3. The table shows three objective mean columns and one subjective mean column. They are:

Mean Time: Shows the average time for a trial in seconds.

Mean Close: Shows the percentage of the controlled curve on or near the target curve. Values of 100 percent are possible for any spline type/target curve combination. This value reflects the amount of red eliminated from the controlled curve.

Mean Exact: Shows the percentage of the target curve that has been exactly matched by the controlled curve. This value reflects the amount of green on the final target curve. In the authors opinion, values above 50 percent can be regarded as good matches.

Mean Rating: Shows the average subjective self rating of the match quality performed after each trial. The scale was from 1 (poor) to 7 (very good).

The table shows means for cusp and non-cusp curves for each spline type. These means were calculated during an analysis of the individual subject means. The data for the analysis is presented in Appendix D: Subject Means.

The analysis was an analysis of variance performed on the subject means for each of the dependent measures mentioned above: Time, Close, Exact, and Rating. Two factors, the main effects in the analysis, corresponded to the conditions that all subjects were tested on. In this experiment, the main effects were the spline types for controlled curves and cusp/non-cusp for target curves. Moreover, there was an additional factor showing the interaction between the two main effects. For each spline type and dependent measure the interaction factor uses the difference between cusp curve means and non-cusp curve means. It shows whether this difference is significant between spline types.

The interaction shows whether the difference, between the spline type averages, depend on the type of curve (cusp/non-cusp).

For each factor, main effect or interaction, two components from the analysis of variance were calculated: an F-value (along with the associated degrees of freedom)

[Ostle 75] and a ρ value showing the confidence level for that result. The usual confidence level of 0.05 or less was used to decide which effect was large enough to be considered statistically significant.

Spline Type	Mean Time C—NC	Mean Close C—NC	Mean Exact C—NC	Mean Rating C—NC
Bs10	146.36 — 117.14	94.78 — 96.34	48.41 — 53.10	4.91 — 5.56
Bsp7	157.69 — 109.39	91.60 — 96.62	39.41 — 46.84	4.14 — 5.09
Beta (tension)	165.94 — 114.91	93.93 — 96.73	41.41 — 46.06	4.34 — 5.14
Ratb (tension)	183.18 — 115.98	90.32 — 95.95	36.65 — 44.20	4.04 — 5.15

Table 5.3: Tension Experiment Means.

The most obvious effect from the analysis of means in Table 5.3 was the cusp/non-cusp factor, which showed a difference between cusp/non-cusp target curves regardless of the controlled spline type. This effect was significant over all measures:

- Time (F(1,23) = 99.24, $\rho < 0.0001$) with the cusp average 163.29 seconds to the non-cusp average 114.36 seconds;
- Close (F(1,23) = 31.17, $\rho < 0.0001$) with the cusp average 92.76 percent to the non-cusp average 96.41 percent;
- Exact (F(1,23) = 100.89, $\rho < 0.0001$) with the cusp average 41.47 percent to the non-cusp average 47.55 percent;
- Rating (F(1,23) = 130.02, $\rho < 0.0001$) with the cusp average 4.37, on 1 to 7, to the non-cusp average 5.24, on 1 to 7.

Clearly matching curves with a cusp was more time consuming and resulted in poorer matches. The fact that cusp/non-cusp curves always produced a large significant difference was expected.

The spline type factor was not as systematic. This factor showed significant differences between the controlled spline types, regardless of cusp/non-cusp target curves, for two measures: Exact and Rating. The Exact measure showed differences between the spline types with a confidence level of $F(3,69) = 16.84$, $\rho < 0.0001$ in which the ordered means were as follows: Bs10 (50.76), Beta (43.74), Bsp7 (43.13) and Ratb (40.43). The Rating measure also showed differences between the spline types with a confidence level of $F(3,69) = 13.93$, $\rho < 0.0001$, in which the ordered means occurred in the same sequence of Bs10 (5.24), Beta (4.74), Bsp7 (4.62), and Ratb (4.60). Note that in both cases the Bs10 results were substantially better than for the other spline types.

There were also two dependent measures for which the interaction of the two main effects was significant. These measures, Time and Close, showed that some spline types fared poorer than others in the degradation from non-cusp to cusp curves. The Time measure indicated the degradation differences (Δ t) with a confidence level of $F(3,69) = 10.13$, $\rho < 0.0001$ in which the ordered spline type degradations in seconds are Bs10 (Δ t = 29.22), Bsp7 (Δ t = 48.30), Beta (Δ t = 51.03), and Ratb (Δ t = 67.20). The Close measure showed the degradation differences with a confidence level of $F(3,69) = 5.13$, $\rho < 0.003$ and the degradation of the means (in percent points) of Bs10 (1.56), Beta (2.80) Bsp7 (5.02), and Ratb (5.63).

Across all significant measures from the analysis of variance relating to the means in Table 5.3, the Bs10 fared consistently better than the other spline types and the Ratb fared consistently worse. However, the interaction factor indicated that there should be further analyses. The most obvious were separate analyses on the four spline types for cusp curves only, then for non-cusp only curves. The current interaction only showed that there is some difference between spline types in the performance of cusp curves compared to the performance of non-cusp curves. A separate cusp curve analysis was performed to check whether the previously presented spline rankings were consistent, i.e, Bs10 best and Ratb worst. Another possible analysis stems from the fact that both non-tension splines fared better on Time. This indicated that performing an analysis on tension splines versus non-tension might give useful information.

5.3.2 Separate Cusp/Non-cusp Analyses

First was the analysis of the spline types for cusp curves only, which strengthened the above general indication that Bs10 was best and Ratb worst. For example, there were three significant dependent measures: Time, Exact, and Rating. The Time measure showed a spline type difference with a confidence level of $F(3,69) = 3.50$, $\rho < 0.02$ in which the means were Bs10 (146.36), Bsp7 (157.69), Beta (165.94), and Ratb (183.18). The difference between spline types as indicated by the Exact measure had a confidence of $F(3,69) = 18.01$, $\rho < 0.0001$ in which the means were Bs10 (48.41), Beta (41.42), Bsp7 (39.41), and Ratb (36.65). The Rating measure also showed a difference between spline types with a confidence of $F(3,69) = 13.11$, $\rho < 0.0001$ and the ordered means Bs10 (4.91), Beta (4.35), Bsp7 (4.15), and Ratb (4.04). Unexpectedly, the tension splines did not fare well on the cusp curves. Another curious observation is that the Beta and Bsp7 spline types have switched rankings on different measures suggesting that these spline types have fared similarly on average.

The spline types for only non-cusp curves were not expected to produce significant results as all spline types should perform basically the same on the simpler non-cusp curves. However, this was not the case, as there were two dependent measures, Exact and Rating, with significant differences. The Exact measure showed a difference between the spline types with a confidence of $F(3,69) = 10.43$, $\rho < 0.0001$ in which the ordered spline types means were Bs10 (53.10), Bsp7 (46.84), Beta (46.06), and Ratb (44.20). The Rating measure, with a confidence of of $F(3,69) = 5.52$, $\rho < 0.002$, produced a heretofore unseen ordering of spline type means of Bs10 (5.56), Ratb (5.15), Beta (5.14), and Bsp7 (5.09). This is the first time (and only time) that Ratb has not held the last position. However, the non-cusp curve results are not as meaningful as the cusp curves analysis results since tension should not be required for the non-cusp curves. In any case the largest and best difference between spline types is again the Bs10.

5.3.3 Tension/Non-tension Analyses

In order to study the issue of tension versus non-tension two more analyses were performed: one for cusp curves, one for non-cusp curves. In both analyses, for all statistically significant results, non-tension fared consistently better than tension. For instance in the study with only cusp curves, non-tension fared better in three of the dependent measures: Time, Exact, and Rating. The Time measure showed differences between tension and non-tension with a confidence level of ($F(1,23) = 7.45$, $\rho < 0.012$) in which the non-tension average of 152.03 seconds was substantially faster than the tension average of 174.56 seconds The Exact measure showed a significant difference between tension and non-tension with a confidence level of $F(1,23) = 25.21$, $\rho < 0.0001$. This measure showed that non-tension (43.91) was 4.88 percentage points better than tension (39.03). The final measure, Rating, continued this difference pattern of non-tension (4.53) ranking better than tension (4.19) in which the difference had a confidence level of $F(1,23) = 8.48$, $\rho < 0.0079$.

Moreover, the analysis of tension/non-tension for non-cusp curves only again produced a significant measure, although no difference had been expected. The Exact measure showed the same difference pattern of non-tension (49.97) performing, on average, 4.84 percentage points better than tension (45.13) splines, in which the difference had a confidence of $F(1,23) = 14.66$, $\rho < 0.0009$. Hence over both cusp and non-cusp curves, non-tension performed better than tension; even though non-tension was expected to perform worse with the cusp curves and show no difference in the non-cusp curves.

5.3.4 Pairwise Spline Analyses

The analyses presented thus far were concerned with the four separate spline types and can only answer one of the questions presented at the beginning of this section, namely how the two tension splines (Beta, Ratb) perform against a non-tension spline with more control vertices (Bs10). Since the only consistent pattern across all relevant analyses was that the Bs10 had the best results in any of the significant measures, it seems clear that Bs10 performed better than both the Beta and Ratb tension splines.

To check this, two pairwise analyses were done: one between Bs10 and Ratb and one between Bs10 and Beta. Both analyses considered cusp and non-cusp curves. In the analysis between Bs10 and Ratb, there was a significant difference in the interaction between spline types and target curve types for all measures of:

- Time: ($F(1,23) = 37.55$, $\rho < 0.0001$) in which Bs10 worsened by 29.21 seconds from non-cusp to cusp while Ratb worsened by 67.20 seconds;
- Close: ($F(1,23) = 21.48$, $\rho < 0.0002$) in which Bs10 worsened by 1.55 percentage points from non-cusp to cusp while Ratb worsened by 5.21 percentage points;
- Exact: ($F(1,23) = 6.17$, $\rho < 0.021$) in which Bs10 worsened by 4.69 percentage points from non-cusp to cusp while Ratb worsened by 7.56 percentage points;

- and Rating: ($F(1,23) = 7.71$, $\rho < 0.011$) in which Bs10 worsened by 0.65 from non-cusp to cusp while Ratb worsened by 1.11.

Moreover, there were significant differences, between the spline types regardless of target curve type, in two measures Exact ($F(1,23) = 33.29$, $\rho < 0.0001$) and Rating ($F(1,23) = 25.26$, $\rho < 0.0001$). Both measures showed the Bs10 clearly the best over Ratb by 10.33 percentage points and 0.64 on 1 to 7 respectively. Clearly the Bs10 spline can be said to be superior, in this study, to the Ratb tension spline.

In the analysis between Bs10 and Beta there was only one measure in which there was a significantly different interaction factor. This measure was Time with confidence of $F(1,23) = 13.23$, $\rho < 0.0014$, and in which the Bs10 worsened by 29.21 seconds from non-cusp to cusp while Beta worsened by 51.03 seconds. However, there were additional significant differences between the two spline types regardless of target curve type in two measures: Exact and Rating. The Exact measure showed a difference between Bs10 (50.76) and Beta (43.74) with a confidence of $F(1,23) = 18.03$, $\rho < 0.0004$. The Rating measure showed a difference between Bs10 (5.24) and Beta (4.74) with a confidence of $F(1,23) = 16.85$, $\rho < 0.0005$. Thus the Bs10 is superior to both of the tension splines, Ratb and Beta, within the current experiment.

The next most obvious pattern is in the comparison of the two tension splines. In all cases, except for the previously discussed analysis of non-cusp curves in the Rating measure, in which the Ratb (5.15) performed 0.01 better than Beta (5.14), the Beta fared consistently better than the Ratb. This is strong evidence that the Beta tension is better in interactive curve design than the type of tension supplied by Ratb.

Once again this was checked by a analysis of variance performed between the two tension spline types. There were significant differences between Beta (worsening by 2.80 percentage points) and Ratb (worsening by 5.21 percentage points) in the interaction between spline type and curve type in the Close measure with a confidence of $F(1,23) = 4.82$, $\rho < 0.039$. Another significant difference ($F(1,23) = 4.36$, $\rho < 0.049$) in the interaction factor of the Rating measure, had Beta worsening by 0.80 from non-cusp to cusp while Ratb worsened by 1.11. There was one significant difference between the two tension spline types regardless of target curve type: Exact. It showed a difference between Beta (43.74 percentage points) and Ratb (40.43 percentage points) with a confidence of $F(1,23) = 6.42$, $\rho < 0.019$.

The last question that needs to be addressed is the comparison of the two tension splines (Beta, Ratb) against the non-tension spline with the same number of control points (Bsp7). When comparing the Ratb and the Bsp7, the Bsp7 is consistently better in all analyses involving the separate spline types except for one. This one is again the analysis of the non-cusp curves in the Rating measure, where the Ratb (5.15) performed better than the Bsp7 (5.09) by 0.06. Since the ranking in all other relevant analyses was Bsp7 always better than Ratb, and obviously so when looking at the cusp curves, the Bsp7 can be considered as slightly better spline than the Ratb.

To check this an analysis of variance was performed between the Bsp7 and the Ratb. There was only one result which indicated a significant difference between the two spline types. This was the Time measure in the interaction of spline types and target curve types. It showed ($F(1,23) = 4.98$, $\rho < 0.036$) the Bsp7 worsening by 48.30 seconds from non-cusp to cusp curves while the Ratb worsened by 67.20 seconds. This one significant difference upholds the the indication of Bsp7's slight advantage over Ratb.

Since there was no clear distinction between the Bsp7 and Beta spline in any previous analyses, two pairwise analyses were performed to see if one type had a clear advantage. The analyses, like previous pairwise analyses, used both cusp and non-cusp curves and showed a significant difference for the interaction factor in the Exact measure of $F(1,23) = 4.38$, $\rho < 0.048$. The results showed Beta reduced by 4.65 percent from non-cusp to cusp curves while the Ratb worsened by 7.43 percent over the same transition. Since none of the other measures were significant, these results suggest only a marginal superiority for the Beta over the Bsp7.

Summing all the above analyses, the Bs10 was clearly superior to the Beta, which was marginally better than the Bsp7, which was slightly better than Ratb. The general result was that tension splines fared worse than non-tension splines.

5.3.5 Comment Form Results

Other important results have been derived from the subject's comment forms. The spline type preferred by subjects was calculated using the spline preference answers after the last session. When one particular spline was indicated, the count for that type/question was augmented by one; a null answer (in one case) augmented all types for the appropriate question by 0.25 and dual preferences augmented the indicated types/question by 0.5. This tally is presented in Table 5.4. As can be seen in the table, Bs10 was the favorite.

	Beta	Bs10	Bsp7	Ratb
Q1	5	13	2.5	3.5
Q2	4	12	3.5	4.5
Q3	3.75	10.25	5.75	4.25
Q1: "In which session did you find control of the curve easiest?"				
Q2: "In which session did you think that you were most successful in your final matches?"				
Q3: "Which session did you enjoy the most?"				

Table 5.4: Subject Preference.

5.4 Discussion

From the general observations on the subject's match methodology and the subject's comments, it is obvious that the instructions need to be changed slightly to indicate how to perform an exact match. The word "overlay" does not necessarily imply matching the target curve length (see help page 2 in Appendix A). One sentence explicitly stating the match size requirement should be added to the instructions.

The resulting answers to the three basic questions make intuitive sense:

- Tension is slower and does not provide an obvious gain in match quality. This can be explained by the complexity of the tension technique. Intuitively subjects are overwhelmed by the 14 degrees of freedom offered by the tension splines and by the necessity of having to use two different spline curve manipulation techniques.
- The Bs10 spline fares better than the tension splines since this spline can match all possible curves with only three extra degrees of freedom.
- The Beta2 spline seems a superior technique in comparison to the rational B-spline. This is attributed to the less intuitive traits of stretching and stiffness effects observed in the rational B-spline.

The poor showing of tension argues against providing tension in interactive curve design, and in favour of allowing the use of additional control vertices.

Note that although there were numerous significant results obtained from the experiment, the subjects were all novice spline users and the tension splines were only compared against the B-spline technique. More experiments are needed with different control groups, different target curves and a modified tension interface.

One aspect of the experiment that was not discovered until the experiment was completely finished was that translating the target and controlled curves altered the position of a few pixels. This is caused by the Iris hardware truncating the floating point values of the control vertices to the integer values of the iris display. This problem is considered trivial since first, none of the subjects mentioned this problem, second, all subjects were affected equally, third, the number of total pixels affected was small (on the order of one pixel in a hundred), and fourth, the pixels changed randomly so that the final pixel count is expected to be quite close. However, this problem should be studied and fixed for future experiments. Additional future considerations are discussed in the next section.

Chapter 6

Summary and Direction

Overall the thesis can be summarized in two main points. Firstly, in relation to the experiment code there is more work to be done. This should be obvious from the experiment control presented in Chapter 2: System Organization, and from the fact that the data structures presented in Section 3.1: Spline Objects, would be more appropriately dealt with in a language like C++. Secondly, the tension experiment performed showed conclusively that tension did not perform better than non-tension.

More generally, the tension experiment provided a platform for exploring the value of tension in interactive curve design. It also instigated a major rewriting of the experiment system code. However, there is a vast amount of work remaining, both in performing more experiments and in reworking the code to develop an even more generalized spline testbed.

Some of the code enhancements that might be considered are:

- Generating an additional intermediate format for playback information. This information would be somewhere between the current binary playback data and the text representation of the experiment results. Doing so would facilitate a wider range of statistical analysis, if more is needed.

- Providing a standardized statistical analysis package specific to the shape matching system for quick experiment analysis. Another possibility for reducing the time of analysis is to provide routines for massaging the experiment results into a form appropriate for an existing standard analysis package.
- A more robust experiment definition language to avoid the need for hardcoded information in `Setup` and `Control` programs.
- Breaking the session control and experiment control aspects of the `Control` program into two separate entities. This would complete the desired modeling of the stages of an experiment.

Possible experiment enhancements are:

- A few ideas for enhancing the tension interface were gained from watching the playback data generated by the experiment subjects. For instance, it might be helpful to always display the tension for each control vertex so that subjects do not have to click on the vertex to check the amount of tension. Moreover, the tension interface could be entered by clicking within a tension ring surrounding but not in the control vertex. This would eliminate the need to differentiate a mouse button. One final consideration would be to change the tension to react slowly when the cursor is close to the control vertex and quickly when it is far from the control vertex. These changes have been made to the current tension interface; whether they will aid the user can best be determined by a second experiment.

- Including a recorded match demonstration as a `Help` demo. This would help make clear to a subject how the controlled curve is manipulated and exactly what is meant by a match.
- Adding a clock to the experiment interface to provide the subject with match time feedback.
- Running experienced spline subjects on the tension technique and comparing their results against the results reported here for novice subjects.
- Running another tension experiment in which only positive tension is allowed. This would test the usefulness of positive versus negative tension.

Some more general code possibilities are:

- Porting the code to an object oriented language. This would provide an excellent chance for further code improvement.
- Generalizing the system further to reduce the number of code changes necessary to run experiments. Eventually the entire experiment system might have a graphical interface that would lead the experiment administrator through the necessary experiment set up, run and analysis steps.
- Including the ability to handle spline surfaces. This would require new interaction techniques and open up an entirely new research area.

Concluding, the more general experiment-related questions are:

- How do workstation parameters affect spline interaction? I.e., changing the mouse response time and the refresh rates.

- What is the best interaction methods for spline curves? One might study interaction techniques rather than spline types as, for example, by comparing control vertex movement with direct curve manipulation [Bartels 89].
- What is the optimal number of control vertices for given shapes? For example, one might allow the subject to add and delete control vertices during a match.
- Can the manipulation of spline surfaces be tested? If so, which spline type is best suited to forming surfaces and would tension be useful in shaping surfaces?

Bibliography

[Bartels 89] Richard H. Bartels, John C. Beatty, "A Technique for the Direct Manipulation of Spline Curves" *Graphics Interface Conference Proceedings, 1989, p. 33-39.*

[Bartels 87] Richard H. Bartels, John C. Beatty, Brian A. Barsky, *An Introduction to Splines for use in Computer Graphics and Geometrix Modeling*, Morgan Kaufamnn Publishers Inc., 1987.

[Bartels 84] Richard H. Bartels, John C. Beatty, "Beta-Splines With a Difference", Technical Report CS-86-65, University of Waterloo, Ontario, Canada, December, 1984.

[Barsky 83] Brian A. Barsky, John C. Beatty, "Local Control of Bias and Tension in Beta Splines", *Computer Graphics, Siggraph Conference Proceedings, 1983, Vol. 17, Num3, p. 193-277.*

[Bloomenthal 83] Jules Bloomenthal, "Edgè Inference with Applications to Antialiasing", *Computer Graphics, Siggraph Conference, Proceedings, 1983, Vol. 17, Num3, p. 157-162.*

[Bosch 87] Eric G. Bosch, "WorkStation-Based Shape Matching Experiments", M-Math Thesis, University of Waterloo, Ontario, Canada, 1987.

[Dixon 57] Wilfrid J. Dixon, Frank J. Massey, Jr, *Introduction to Statistical Analysis*, McGraw-Hall Book Company Inc., 1957.

[Evans 81] Kenneth B. Evans, Peter P. Tanner, Marceli Wein, "Tablet Based Valuators that provide One, Two, or Three Degrees of Freedom", *Computer Graphics, Siggraph Conference Proceedings, 1981, Vol. 15, Num3, p. 91-97.*

[Field 85] Dan Field, "Incremental Linear Interpolation", *ACM Transactions on Graphics, 1985, Vol. 4, Num1, p. 1-11.*

[Newman 79] William M. Newman, Robert F. Sproull., *Principles of Interactive Computer Graphics, second edition*, McGraw-Hill Inc., 1970.

[Ostle 75] Bernard Ostle, Richard W. Mensing, *Statistics in Research, 3rd edition*, Iowa State University Press, 1975.

[Price 61] Matlack Price, *Foot-High Letters: A Guide to Lettering*, McClelland and Stewart Ltd., 1961.

[Paeth 86] Alan Paeth, "The Im Raster ToolKit Design, Implementation and Use", Technical Report CS-86-65, University of Waterloo, Ontario, Canada, December, 1986.

[Roy 88] Marie-France Roy, "Rational B-Splines: Properties and Applications", MMath Essay, University of Waterloo, Ontario, Canada, 1988.

[SGI 88] Silicon Graphics, Inc., "GT Graphics Library User's Guide", Silicon Graphics, Inc., 1988.

[Stroustrup 87] Bjarne Stroustrup, *The C++ Programing Language*, Addison-Wesley Publishing Company, 1987.

Appendix A

Help Text Pages

The help pages are relatively straightforward since they consist mainly of text. However the few pieces of control information that are embedded can be described as:

- The leading number of pages in the help text. In this case there are eight (8) pages.
- The marker for the end of a page. The pound sign (#) is used for this purpose.
- The control sequence to trigger one of the demonstrations. This is indicated by the letter "e" followed by the number of the demonstration. A "1" indicates that the demo is to remain visible after the user has acknowedged its presence, a "0" otherwise.

The actual help pages are:

```
8
                                                            - Page 1 -

                              University of Waterloo
Computer Graphics Laboratory / Psychology Department Curve-Matching Experiments

        The experiment you are about to participate in is part of an
investigation into the effectiveness of various curve drawing techniques.

        Before you begin the actual experiment, you will be led through a
tutorial that will familiarize you with the computer graphics equipment and the
curve drawing process.
```

Please take your time during the tutorial. It is important that you understand how the computer graphics equipment works. You may adjust the position of any of the devices you will be using during the experiment. If at any time during the tutorial or during the actual experiment you are uncertain as to what is required of you, please feel free to ask the experiment supervisor for assistance.

The primary piece of graphics equipment that you will be using is called a MOUSE and is positioned on a mouse pad in front of the graphics terminal. By moving the mouse over the surface of the mouse pad you can control the position of the red arrow, called a CURSOR, on the graphics display. Note that the movement of the mouse is not absolute, but is instead relative to its previous position while in contact with the pad. Thus you can lift the mouse off the mouse pad, place it at a new location on the pad, and then continue moving the cursor.

During this tutorial the mouse will be used to PICK certain objects on the graphics display. To pick an object, position the cursor within the object and then press any of the three buttons (called MOUSE BUTTONS) on the top of the mouse. For the purpose of this experiment, the left and middle mouse buttons are treated identically. The right mouse button is for the most part identical to the other two, but sometimes has a special function which will be described later.

The boxes visible on the left-hand side of the graphics display represent a table of contents for the introductory pages you are now reading. Each box corresponds to a different page of this introduction. There are only two boxes visible at the moment: as you read additional pages, additional boxes will become visible.

Note that the boxes are outlined in different colours. A red outline indicates that the corresponding page is currently being read, while a green outline indicates that the corresponding page has not yet been read. Once a page has been read it will be outlined in white. Look, for example, at the currently visible boxes: the first is outlined in red because it corresponds to this page (the one you are now reading); the second box is outlined in green because it has not yet been read.

When you are finished reading a page, you move on to another by using the mouse to pick a box from the column of visible boxes on the left of the graphics display. You can review previously read pages by picking one of the white-outlined boxes, or you can continue with new material by picking the green-outlined box. If you pick the box outlined in red, nothing will happen, since the page picked is the page currently displayed. For example pick the red outlined box labelled Page 2 to read the next page.
#

- Page 2 -

```
        This particular experiment is divided into four independent sessions.
Over the course of the four sessions you will be using and evaluating each of
four curve drawing techniques, one per session.

        Each session is comprised of twenty four trials (what constitutes a
trial will be explained shortly).  The twenty four trials are separated into
four initial practice trials and twenty recorded trials.  The transition
between practice trials and recorded trials is well marked.  In addition, the
number of trials remaining is displayed in the lower-left corner of the
graphics display.

        During each trial you will be presented with two curves drawn inside
the MATCH WINDOW on the graphics display.  The brown curve will remain the same
throughout the trial and is called the TARGET CURVE.  The shape of the red
curve can be changed, and it is therefore called the CONTROLLED CURVE.  Several
small squares or circles are displayed along with the controlled curve.  These
symbols (called CONTROL POINTS) are connected by straight lines (which form the
CONTROL POLYGON).  Your task during each trial is to manipulate the controlled
curve so that it matches the target curve as closely as possible in a short
amount of time.

        A match is accomplished by causing the controlled (red) curve to exactly
overlay the target (brown) curve. It may not be possible to match the two curves
exactly, but you should try to get as close as you can within a few minutes.
e 1 1

#
                                                             - Page 3 -

        If you press and drag a mouse button when the cursor is within the
match window, but outside any of the control points, the controlled and target
curves can be repositioned, as a unit, anywhere within the match window.
You may sometimes find this convenient.  For a better response only the
target curve is drawn during the translate even though both curves will
be identically translated when the mouse button is released.

        Pressing a mouse button outside of the match window toggles the
visibility of the control points and control polygon.  You may find it useful
to turn them off momentarily when they obscure a portion of the curve.

        Try using the cursor to translate the entire curve now.  Then try
using the cursor to toggle the visibility of the control points and
control polygon.  When you are finished pick the box labeled STOP.

        Note that the demo window you see is a small version of the experiment.
This help session is only meant to teach you how to work the experiment.  There
```

```
will be a more appropriate time to practice an actual match after this help
session is over.
e 2 0

#
                                                             - Page 4 -

        You can move a control point by positioning the cursor within the symbol
representing it (a circle or square), and then holding down the left or middle
mouse button while you move the mouse.  Pressing a mouse button causes the red
cursor and the control polygon to disappear, confirming your selection.  When
you release the mouse button, the red cursor and the control polygon reappear,
indicating that you are no longer moving that control point.

        As you move a control point, the controlled curve changes shape.
The manner in which the controlled curve responds to control point movement
is determined by the curve drawing technique in use.  Each session uses a
different curve drawing technique which remains constant throughout the session.

        Try using the cursor to move a few control points now and when you are
finished pick the box labeled STOP.
e 2 0

        You may have already noticed that it is NOT possible to pick a control
point and move it outside the match window, but that it IS possible to move
one or both curves so that a control point is positioned outside the match
window.  While there is nothing to prevent you from doing this, it should
not be necessary.

For a spline having square control points the right mouse button is
identical in every respect to the other two mouse buttons; for splines
having circles as control points, the right mouse button has a different
purpose, which is explained on the next page.
#
                                                             - Page 5 -

        Splines having circular control points also allow you to alter the
controlled curve's TENSION:  in doing so you will pull the curve towards a
control point or push it away from a control point.  For such curves
selecting a control point with the right mouse button will allow you to
alter the tension associated with that control point.

        The control polygon is turned off while you are altering its tension,
just as it is when you change its position, but there are some additional
effects: the cursor remains visible, and an additional circle appears, together
with a line connecting the cursor to the center of the control point.  To change
the tension, move the cursor in a circular motion around the control point.  A
```

```
clockwise motion increases the tension, pulling the curve towards the control
point.  A counter-clockwise motion decreases the tension, pushing the curve
away from the control point.

        An arc on the outer circle indicates the approximate level of tension.
All control points start with no arc, indicating zero tension. A bright arc
indicates that you have applied positive tension, pulling the curve towards the
control point; a dim arc indicates that you have applied negative tension,
pushing the curve away from the control point. If an arc of neither brightness
is visible then no tension is currently being applied at that control vertex.

        Try using the cursor to move a few control points and adjust their
tension now.  When you are finished pick the box labeled STOP.
e 5 0

        There are a few things to take note of:

        You may apply tension at any number of control points.
Also note that it is easier to see the effect of a change in tension at
a control point when the control point does not overlay the control curve.

        To achieve a finer degree of control when adjusting the tension move
the cursor further away from the selected control point.

        One full rotation around the control point, starting from neutral
tension, will bring the tension to its maximum or minimum depending on the
direction travelled.  Maximum or minimum tension is indicated by the bright or
dim arcs forming a complete circle.  You will not be able to change the tension
past its maximum or minimum.

        Once again, only splines whose control points are represented by
a circle have the tension attribute.  You can also tell the difference by
selecting a control point with the right mouse button and watching the
program's response: if the paraphernalia for changing tension doesn't
appear, then you aren't working with a tension spline.
#
                                                              - Page 6 -

        At the end of every trial you will be asked to rate the quality of
your match.  The rating scale has seven values, from POOR through MEDIUM
to VERY GOOD.  It is suggested that during the course of the experiment you
make use of the entire scale by tailoring the rating range to your own
personal performance, so that your best match is VERY GOOD and your worst
is POOR.

        To make a selection on the rating scale, move the mouse so that
the cursor is within the appropriate box.  As you move the cursor along the
```

```
scale, the currently selected box is temporarily highlighted.  When you are
satisfied with your selection, press any mouse button while the desired
box is selected to record your rating.  Try this now.
e 3 0

#
                                                              - Page 7 -

        By now you should be familiar with the mouse and how it is used to
pick objects.

        During a trial there are three screen configurations: the START screen,
the INTERACTION screen, and the RATING screen.  In the starting configuration
you are shown an empty window and a box labeled START.  When you are ready
to start a trial, pick the box labeled START.  Try this now.
e 4 1

        The display then changes to the interaction screen.  During this
stage you are to manipulate the controlled curve so that it matches the
target curve as closely as possible.  A close match is determined as follows:
when you move the controlled curve over the target curve it will change colour.
GREEN means it is exactly over the target curve; GREY means that it is very
close.  You should continue to try matching until there is no red remaining
in the target curve, or until you find you cannot improve the quality of
your match.  When you are satisfied with your match, or decide that you
cannot improve it, pick the box labeled STOP.  Try this now.  Note, that a
a good match may be difficult due to the small size of the demo.
e 2 1

        The rating scale is now superimposed on the interaction screen.  Make
your rating selection by moving the cursor within the appropriate box and
then pressing any mouse button.  Try this now.
e 3 0

        One important thing to note is that it is not possible to go back to
the interaction screen once you have selected the rating screen, and similarly
it is not possible to go back to the start screen once you have selected the
interaction screen.
#
                                                              - Page 8 -

        Congratulations!  You have just completed one entire practise trial.
If you do not feel prepared to begin the experiment now, please go back and
review the necessary instruction page(s).

        It is important that you complete all of the twenty trials constituting
a session in one uninterrupted sitting.  If you wish to pause during a session,
```

please do so between trials, and not during a trial.

A good idea would be to summarize to youself the different aspects of the interaction for the experiment. In brief the important things to be understand well are:

- o The presence of square control points means the curve has no tension. (page 3)
- o The presence of circular control points means the curve has tension. (page 5)
- o Pressing any mousebutton outside the match window will toggle the control polygon. (page 3)
- o Pressing any mousebutton inside the match window but not in a control point will move both curves. (page 3)
- o Pressing the LEFT or MIDDLE button inside any control point will move the control point, reshaping the curve. (page 4)
- o Pressing the RIGHT mouse button inside a control point will change the tension, if a tension curve is being displayed, otherwise it will move the control point. (page 4, page 5)
- o A clockwise motion increases tension (pulls the curve closer closer to the control point) while a counterclockwise motion decreases tension (pushes the curve away from the control point). (page 5)
- o A bright arc indicates positive tension while a dim arc indicates negative tension. (page 5)
- o An exact match is shown by GREEN. (page 7)
- o A close match is shown by GREY. (page 7)
- o Match as quickly and as closely as you can. (page 2)

Please ensure that all these points are clear, reviewing the indicated page(s) if necessary.
#

Appendix B

Random Spline Generation

For a number of future experiments it will be desirable to generate random splines of different curve types for use as target curves. Although it was not necessary for the current experiment, such a program was developed and used by Bosch. It was ported and modified and modified by the author to fit with the current spline experiment system.

`GetTarget` is a port and enhancement of Bosch's `Target2` code. It generates random splines using a simple and adaptable interface. It has three main window areas: one for displaying the generated spline, a second for a list of user option buttons and a third to print status information for the user (see Figure B.1).

As can be seen from the diagram a user's options are as follows:

ACCEPT write the viewed curve to a file,

REJECT generate a new curve of the same type,

TOGGLE turn on and off the control polygon of the curve,

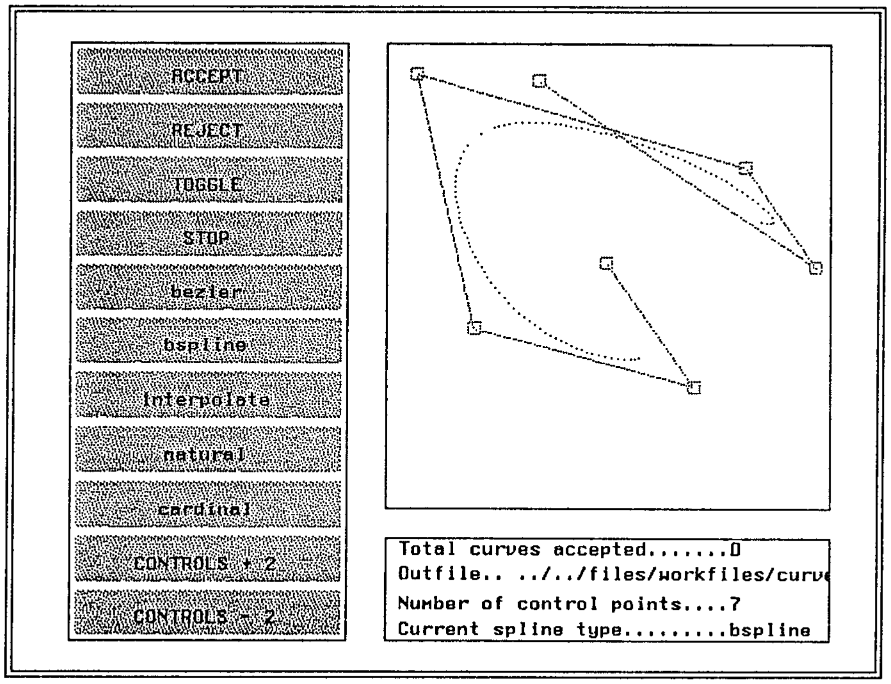
STOP quit the program,

BEZIER generate Bézier curves from now on,

BSPLINE generate B-spline curves from now on,

INTERPOLATE generate C^2–Interpolating curves from now on,

NATURAL generate Natural Cubic splines from now on,



```
Total curves accepted.......0
Outfile.. ../../files/workfiles/curv
Number of control points....7
Current spline type.........bspline
```

Figure B.1: GetTarget Screen.

CARDINAL generate Cardinal curves from now on,

CONTROLS plus 2 add to the present number of control vertices,

CONTROLS minus 2 subtract from the present number of control vertices.

There are currently four lines of feedback which are displayed underneath the curve window and to the right of the button list. The feedback consists of:

- the number of curves that have been written to a file,
- the current spline type,
- the current number of control vertices,
- the output file name.

One of the least favoured changes was the exclusion of accepting a script command file. This could be reinstated for future use. For now the program is useful for generating a file of curves or a single curve, but cannot be used to generate a set of files, each containing a single curve.

Appendix C

Running the Current Experiment

The two main steps for running an experiment are:

1. Create the target curves as explained in Section 4.2: Processing, and put all the letters into the `match/files/curves` directory using the naming convention discussed in Section 2.1: Setup.

2. Create the subject definition file for `Setup`. The current experiment is shown in Figure C.1. Edit `Setup` to produce the hardcoded auxiliary information of window size and `Help` text location. Then compile and run `Setup` once. This will produce the experiment specification files in `match/files/session`. These files are called `exfile1` to `exfile24`.

The next step is to copy the entire match directory over to the Iris. All executables will have been previously compiled on the Iris over NFS links to the Vax 8600.

The remaining step is to replace the contents of the `.Shell` file in the directory `/u/frontend` with:

```
cd Shape_Matching/match/source/bin
ClearIris
Control
QuitIris
```

The `.Shell` file serves as the users login shell in place of the regular unix shell. The experiment runs itself from here on. The one thing to be cautious of is monitoring the amount of playback data generated in the `match/files/playback` directory. When

```
#########################################################################
#
subject 01 bspline-7-(AC) rationalb-7-(BD) beta-7-(AD) bspline-10-(BC)
subject 02 bspline-7-(AC) rationalb-7-(BD) bspline-10-(BC) beta-7-(AD)
subject 03 rationalb-7-(BD) bspline-7-(AC) beta-7-(AD) bspline-10-(BC)
subject 04 rationalb-7-(AC) bspline-7-(AD) bspline-10-(BD) beta-7-(BC)
#
subject 05 bspline-7-(BD) beta-7-(AC) rationalb-7-(BC) bspline-10-(AD)
subject 06 bspline-7-(AD) beta-7-(AC) bspline-10-(BD) rationalb-7-(BC)
subject 07 beta-7-(AC) bspline-7-(AD) rationalb-7-(BC) bspline-10-(BD)
subject 08 beta-7-(BD) bspline-7-(BC) bspline-10-(AD) rationalb-7-(AC)
#
subject 09 bspline-7-(BC) bspline-10-(AC) beta-7-(BD) rationalb-7-(AD)
subject 10 bspline-7-(BC) bspline-10-(AC) rationalb-7-(AD) beta-7-(BD)
subject 11 bspline-10-(AD) bspline-7-(BD) beta-7-(BC) rationalb-7-(AC)
subject 12 bspline-10-(BC) bspline-7-(BD) rationalb-7-(AD) beta-7-(AC)
#
subject 13 rationalb-7-(AC) beta-7-(BC) bspline-7-(BD) bspline-10-(AD)
subject 14 rationalb-7-(AD) beta-7-(BD) bspline-10-(AC) bspline-7-(BC)
subject 15 beta-7-(BC) rationalb-7-(AD) bspline-7-(BD) bspline-10-(AC)
subject 16 beta-7-(BD) rationalb-7-(BC) bspline-10-(AC) bspline-7-(AD)
#
subject 17 rationalb-7-(BC) bspline-10-(BD) beta-7-(AC) bspline-7-(AD)
subject 18 rationalb-7-(BC) bspline-10-(AD) bspline-7-(AC) beta-7-(BD)
subject 19 bspline-10-(AD) rationalb-7-(AC) beta-7-(BC) bspline-7-(BD)
subject 20 bspline-10-(BD) rationalb-7-(AD) bspline-7-(BC) beta-7-(AC)
#
subject 21 beta-7-(AD) bspline-10-(BC) bspline-7-(AC) rationalb-7-(BD)
subject 22 beta-7-(AD) bspline-10-(BC) rationalb-7-(BD) bspline-7-(AC)
subject 23 bspline-10-(AC) beta-7-(BC) bspline-7-(AD) rationalb-7-(BD)
subject 24 bspline-10-(BD) beta-7-(AD) rationalb-7-(AC) bspline-7-(BC)
#
practice 1
experimental 10
#
#########################################################################
```

Figure C.1: Complete Subject Definition File.

this gets large (i.e. greater than 20 megabytes), off load the information to the Vax 8600 and back it up to magnetic tape.

There are a couple of items to be aware of when considering the shape match system programs individually. First is that to create an executable for any program it is necessary to `cd` to the program directory and type `make driver`. Additional example parameter files for the main programs are listed below.

For the `CurveMatch` session information the parameter file is:

```
# the session file
mainwindow 0.0 1.0 0.0 1.0
targetview 0.25 0.75 0.25 0.875
controlview 0.25 0.75 0.25 0.875
startstopview 0.45 0.55 0.1 0.2
ratingview 0.3 0.7 0.1 0.2
```

For the `CurveMatch` trial information the parameter file is:

```
# the trial file
experimenttrials 1
controlspline beta 7
targetspline ../../files/curves/trial_D_01
outdata Time_For_Trial
outdata Control_Rating
outdata Target_Rating
outdata Self_Rating
```

An example of running `CurveMatch` for recording a single trial is:

```
cat ../../files/workfiles/sessionrc.tension
            ../../files/workfiles/trialrc.tension |
                 ./CurveMatch -o ../../files/workfiles/tmp.playback
```

An example of running `CurveMatch` for replaying a single trial is:

```
cat ../../files/workfiles/sessionrc.tension
            ../../files/workfiles/trialrc.tension |
                 ./CurveMatch -p ../../files/workfiles/tmp.playback
```

All executables are assumed to be run from the `source/bin` directory.

For the `Help` program the parameter file is:

```
mainwindow 0.0 1.0 0.0 1.0
textview 0.35 0.95 0.05 0.95
iconview 0.05 0.30 0.05 0.95
textfile ../../files/workfiles/helptext
helpsplines bspline 7 ../../files/curves/help_B
```

An example of running the `Help` program is:

```
cat ../../files/workfiles/helprc | ./Help
```

An example of running the `Setup` program is:

```
cat ../../files/workfiles/setuprc.epl | ./Setup
```

A more complete example of the output produced by `Setup` and used to run the experiment is given below. The `outdata` lines have been removed for brevity. Moreover only the first session is shown. This file is named `exfile1` and the complete example can be found in the `files/session` directory.

```
session 1
  mainwindow 0.0 1.0 0.0 1.0
  targetview 0.25 0.75 0.25 0.875
  controlview 0.25 0.75 0.25 0.875
  startstopview 0.45 0.55 0.1 0.2
  ratingview 0.3 0.7 0.1 0.2
  helpfile ../../files/workfiles/helptext
  helpsplines rationalb 7 ../../files/curves/help_B
  practicetrials 4
    practicesplines rationalb 7 ../../files/curves/practice_C_01
    practicesplines rationalb 7 ../../files/curves/practice_D_01
    practicesplines rationalb 7 ../../files/curves/practice_A_01
    practicesplines rationalb 7 ../../files/curves/practice_B_01
  experimenttrials 20
    trialsplines rationalb 7 ../../files/curves/trial_D_05
    trialsplines rationalb 7 ../../files/curves/trial_D_06
    trialsplines rationalb 7 ../../files/curves/trial_D_07
    trialsplines rationalb 7 ../../files/curves/trial_B_01
    trialsplines rationalb 7 ../../files/curves/trial_B_09
    trialsplines rationalb 7 ../../files/curves/trial_D_02
    trialsplines rationalb 7 ../../files/curves/trial_B_03
    trialsplines rationalb 7 ../../files/curves/trial_D_08
    trialsplines rationalb 7 ../../files/curves/trial_B_02
    trialsplines rationalb 7 ../../files/curves/trial_B_10
    trialsplines rationalb 7 ../../files/curves/trial_D_09
    trialsplines rationalb 7 ../../files/curves/trial_B_05
    trialsplines rationalb 7 ../../files/curves/trial_B_06
    trialsplines rationalb 7 ../../files/curves/trial_B_07
    trialsplines rationalb 7 ../../files/curves/trial_D_03
    trialsplines rationalb 7 ../../files/curves/trial_D_01
    trialsplines rationalb 7 ../../files/curves/trial_B_08
    trialsplines rationalb 7 ../../files/curves/trial_D_04
    trialsplines rationalb 7 ../../files/curves/trial_D_10
    trialsplines rationalb 7 ../../files/curves/trial_B_04
session 2
  mainwindow 0.0 1.0 0.0 1.0
                    .
                    .
                    .
```

Two examples of input target curves are given below. The first example is a "real-life" curve generated from the letter set. The target curves can be found in `files/curves`. In both cases the file begins with the spline type followed by the number of $x - y$ control vertices.

```
bspline 391
0.685938 0.714063
0.684375 0.714063
0.682813 0.714063
0.681250 0.712500
0.679688 0.712500
     .
     .
     .
```

```
  380 lines deleted
        .
        .
        .
0.381250 0.265625
0.381250 0.264062
0.382813 0.262500
0.382813 0.260938
0.384375 0.259375
0.385938 0.257813

natural 7
0.000000 0.000000 0.543312 0.066197 0.447264 0.114594
0.937638 0.244992 0.428478 0.572634
0.171277 0.611180 0.000000 0.000000
```

Appendix D

Complete Experiment Results

Subject	Mean Time C—NC	Mean Close C—NC	Mean Exact C—NC	Mean Rating C—NC
1	132.91—112.86	100.0—100.0	65.90—64.20	5.30—5.90
2	201.08—168.83	99.70—99.60	58.30—64.40	5.40—6.00
3	197.70—148.67	98.20—99.80	50.60—56.20	4.80—5.50
4	153.72—118.05	100.0—100.0	60.80—65.60	5.50—5.90
5	112.16—110.06	89.10—92.90	26.70—32.20	4.90—5.60
6	129.85—109.84	99.00—99.40	50.60—54.30	5.50—6.00
7	153.26—116.71	94.00—95.80	29.80—32.50	5.70—6.20
8	196.97—151.69	100.0—100.0	67.10—71.40	5.10—5.80
9	140.44—116.63	94.80—99.00	45.50—56.60	5.30—6.60
10	135.12—126.94	96.10—99.90	43.30—50.00	4.20—5.50
11	254.79—173.77	99.80—99.80	59.60—68.00	4.00—4.70
12	136.61—117.10	99.11—98.90	49.11—58.30	4.44—5.20
13	142.80—138.88	100.0—99.90	60.30—61.50	5.10—5.20
14	105.42— 87.90	97.20—98.80	41.10—48.00	5.60—6.00
15	61.00— 55.74	100.0—100.0	50.30—60.30	5.20—5.70
16	113.42— 88.01	88.40—96.10	26.60—34.10	4.60—5.80
17	229.82—173.18	97.30—97.80	41.90—39.60	4.50—5.30
18	92.23— 58.20	89.90—97.70	31.10—39.00	4.70—6.10
19	94.66— 95.44	35.10—37.60	32.30—33.00	4.90—4.70
20	139.91—115.18	98.20—99.70	45.00—51.30	4.10—5.40
21	183.28—133.96	99.90—99.90	49.10—47.70	4.30—3.90
22	150.12—117.66	100.0—99.70	64.50—59.60	4.80—3.90
23	105.34— 77.40	99.20—99.90	56.90—59.70	4.30—5.80
24	149.98— 98.77	99.80—99.90	55.50—67.00	5.70—6.80

Table D.1: B-spline 10 Means.

Subject	Mean Time C—NC	Mean Close C—NC	Mean Exact C—NC	Mean Rating C—NC
1	122.10—100.96	98.10—100.0	44.30—55.90	3.90—5.80
2	98.63— 79.36	87.70—99.70	34.30—55.90	2.80—5.20
3	165.07— 93.96	79.00—96.00	22.90—45.50	3.90—5.80
4	234.35—150.73	99.70—100.0	51.70—66.10	3.70—5.60
5	110.65— 77.58	70.70—92.70	19.40—28.40	3.70—5.50
6	191.60—132.66	95.80—99.70	43.90—48.40	5.10—5.70
7	254.46—209.69	97.90—98.20	39.80—36.90	5.60—5.70
8	131.36—122.17	99.90—100.0	59.70—56.70	3.70—3.50
9	137.80—105.62	97.10—99.40	40.70—45.30	5.60—6.20
10	144.37— 88.25	99.80—99.70	43.00—54.30	4.30—4.90
11	250.10—158.54	99.70—100.0	59.70—65.30	3.70—4.00
12	135.02—115.05	96.10—99.00	42.10—49.70	4.10—5.30
13	92.56— 74.68	92.70—96.70	30.90—36.50	3.80—4.60
14	109.45— 92.15	94.56—93.50	35.00—30.90	5.22—4.90
15	207.67—135.62	100.0—100.0	56.00—55.80	4.20—4.30
16	92.84— 80.79	87.80—92.10	24.20—34.70	4.10—5.30
17	301.24—120.52	96.70—99.60	35.80—45.00	3.70—4.90
18	102.39— 66.42	70.40—86.30	22.30—30.30	3.90—5.30
19	129.57— 81.51	61.20—69.30	39.00—38.90	4.90—5.10
20	72.42— 60.07	91.30—99.90	28.80—39.30	3.10—5.40
21	186.33— 99.48	96.70—98.50	30.10—39.50	3.70—4.20
22	227.57—166.32	88.50—98.60	38.40—45.10	3.20—4.20
23	162.65—119.96	97.20—100.0	50.50—59.10	3.90—4.60
24	124.48— 93.16	99.80—100.0	53.40—60.60	5.70—6.20

Table D.2: B-spline 7 Means.

Subject	Mean Time C—NC	Mean Close C—NC	Mean Exact C—NC	Mean Rating C—NC
1	149.42— 97.73	98.00—100.0	52.70—55.50	4.80—5.70
2	162.62—142.82	97.20—99.00	39.70—50.30	4.00—4.10
3	189.99— 95.69	96.40—99.50	37.70—49.50	4.90—5.80
4	161.31—105.27	99.90—100.0	58.70—65.00	4.90—5.70
5	97.00— 63.03	80.10—94.80	26.60—31.20	3.90—5.60
6	185.81—121.20	95.90—99.30	38.70—49.50	4.30—4.90
7	104.09— 92.93	98.40—98.30	37.00—40.30	6.00—6.30
8	164.85—117.45	100.0—100.0	57.00—61.10	4.00—4.50
9	170.44—139.44	87.60—90.40	33.60—31.80	5.20—5.00
10	142.69— 97.18	99.00—99.80	51.89—51.70	4.78—4.80
11	252.08—150.76	99.80—100.0	57.60—63.80	3.60—4.60
12	136.15— 93.15	97.00—98.60	46.00—41.30	4.90—5.40
13	101.34— 61.12	94.00—98.70	32.70—40.20	3.60—4.90
14	173.94—163.41	71.60—87.30	25.90—30.90	4.00—5.50
15	199.46— 99.60	100.0—99.60	47.60—49.80	4.00—4.60
16	119.16— 94.46	90.60—81.10	26.90—27.20	4.60—5.10
17	236.44— 98.63	98.30—100.0	44.90—53.70	3.80—5.30
18	155.71— 90.25	82.30—88.00	24.10—27.80	4.50—5.20
19	157.30—181.06	88.50—90.60	35.70—31.60	5.50—5.70
20	127.73—104.74	90.90—97.70	31.90—43.40	4.00—5.40
21	182.77— 87.22	97.00—99.70	37.90—39.00	3.90—4.40
22	161.77—105.03	93.40—99.40	46.70—54.40	3.10—4.30
23	236.09—187.33	99.60—100.0	54.60—60.90	3.30—5.10
24	214.49—168.38	98.90—99.70	47.90—55.60	4.70—5.50

Table D.3: Beta Spline Means.

Subject	Mean Time C—NC	Mean Close C—NC	Mean Exact C—NC	Mean Rating C—NC
1	184.92—134.50	91.50—96.00	36.40—37.00	5.00—5.10
2	150.74— 87.99	86.90—99.10	31.90—44.60	3.20—5.10
3	177.22—109.74	89.40—96.70	30.30—36.40	4.10—5.30
4	305.82—165.78	95.80—99.90	46.30—58.50	3.40—4.60
5	107.29— 74.81	73.90—87.90	19.10—25.40	3.80—5.60
6	168.51—120.62	95.40—97.00	37.90—39.10	5.10—5.40
7	137.46— 85.46	96.70—99.10	34.50—39.80	6.00—6.30
8	309.55—207.48	98.40—100.0	55.90—59.50	4.70—5.60
9	250.08—164.23	74.80—84.20	23.90—30.50	3.20—4.40
10	154.32—105.20	98.50—99.70	42.60—53.60	4.80—4.90
11	246.21—165.56	96.50—98.90	46.30—49.70	4.30—4.60
12	129.45— 78.23	92.00—99.70	34.80—51.90	3.70—5.10
13	121.62— 67.21	95.10—99.10	38.10—43.90	3.60—5.20
14	112.12— 65.46	86.50—96.70	29.90—45.20	4.00—5.70
15	152.72— 69.90	99.40—100.0	44.20—49.60	3.70—4.30
16	215.12—194.91	59.20—69.90	14.90—20.10	3.40—5.00
17	235.00—159.94	97.50—99.70	35.80—43.90	3.90—5.10
18	151.37— 63.31	90.80—94.60	28.50—35.70	4.00—5.60
19	192.14—137.95	74.30—86.00	36.70—40.60	4.60—5.50
20	93.90— 70.34	94.50—99.80	36.90—36.60	4.10—4.90
21	151.03—110.17	97.90—99.60	33.30—43.30	3.40—4.80
22	233.57— 92.29	98.11—99.70	49.22—58.70	3.44—4.40
23	171.09—100.36	98.20—99.90	50.50—56.20	3.80—5.00
24	245.07—152.09	96.30—99.70	41.60—61.10	3.80—6.20

Table D.4: Rational B-spline Means.