Drawing and Animation Using Skeletal Strokes

Siu Chi HSU
Cambridge University Computer Lab
sch@cl.cam.ac.uk

Computer Science Department†,
Chinese University of Hong Kong, Shatin, Hong Kong
schsu@cs.cuhk.hk
†: current correspondence address

Irene H. H. LEE
Creature House Inc.

School of Music,
Hong Kong Academy for Performing Arts

ABSTRACT

The use of *skeletal strokes* is a new vector graphics realization of the brush and stroke metaphor using arbitrary pictures as 'ink'. It is based on an idealized 2D deformation model defined by an arbitrary path. Its expressiveness as a general brush stroke replacement and efficiency for interactive use make it suitable as a basic drawing primitive in drawing programs as well as windowing and page description systems. This paper presents our drawing and animation system, 'Skeletal Draw', based on skeletal strokes. The effectiveness of the system in stylish picture creation is illustrated with various pictures made with it. Decisions made in the handling of sub-strokes in a higher order stroke and recursive strokes are discussed. The general anchoring mechanism in the skeletal stroke framework allows any arbitrary picture deformation to be abstracted into a single stroke. Its extension to piecewise continuous anchoring and the anchoring of shear angle and stroke width are explained. We demonstrated how this mechanism allows us to build up powerful pseudo-3D models which are particularly useful in the production of 2½D cartoon drawings and animation. Animation sequences have been made to illustrate the ideas, including a vector graphics based motion blurring technique.

CR Categories and Subject Descriptors: I.3.2 [Computer Graphics]: Picture/Image Generation - Display algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction techniques.
Additional Keywords: Non-photorealistic rendering, stylish strokes, skeletal strokes, vector graphics, deformation, psuedo-3D model, animation, fractals.

1 INTRODUCTION

We are proposing *skeletal strokes* as a new vector based drawing primitive for general stroking and picture specification. With the skeletal strokes framework, users are freed from the dependence on the scanner to input expressive brush strokes for further processing; they can create these strokes interactively on the computer. Compared with previously proposed expressive strokes, the skeletal stroke framework has more general applications. It has, for example, an anchoring mechanism which could be applied to 2½D animation. The high level abstraction of skeletal strokes makes it particularly attractive in terms of storage size and transmission efficiency. Its efficient implementation is favourable for incorporation into windowing systems and page description languages.

We shall demonstrate the various techniques with our drawing and animation system, Skeletal Draw, which uses skeletal strokes as the basic primitives. An animation sequence has also been made to show the effectiveness of creating 2½D animation using pseudo-3D models.

2 STROKES, SKELETONS AND ENVELOPES

The constant thickness stroke is still the basic drawing primitive provided by most commercially available vector based drawing packages[34,39]. To create pictures like figures 2, 3, 17 and 19 with these packages usually involves scanning in an original copy (which has already been created on paper) or tracing out its outlines manually or automatically. Indeed, many impressive illustrations that fill the pages of manuals of many market-leading packages are there to demonstrate the package's automatic tracing or photo-retouching capabilities.

2.1 What is a Stroke?

Useful as they are for simple designs, constant thickness strokes cannot be compared to more general strokes for convenience and expressiveness. While it is possible to trace out the outlines of a brush trail every time (figure 1), this painstaking drawing method no doubt is a hindrance to the aspiring artist. It is also doubtful whether pictures of reasonable complexity can be managed if all the strokes are traced out this way — consider modifying a few strokes in Figure 3 if the strokes are polygons. In fact, one would not regard such a drawing method as 'drawing with strokes'. If a picture is drawn with strokes, the effects of each stroke should be apparent after a single application of the brush. In this sense, the constant thickness stroke provided by most drawing programs and the bitmap brush (whether antialiased or not) available in paint programs are both considered to be computer strokes. We shall discuss the deficiencies of each shortly.

Figure 1: Tracing out a stylish brush trail with 'primitive' strokes. The letter 'S' in this illustration is drawn with one single skeletal stroke application.

2.2 Skeleton verses Envelope

Many drawing/2D warping packages allow the user to specify the deformation of a picture by reshaping an envelope around the undeformed picture. This technique has been advocated for applications in 2D animation in as early as the 70's[15] and is still widely used in the latest systems[32]. A coordinate system transformation based on envelope deformation is quite different from a skeleton based one. The former is much more well defined as the coordinates within an envelope can usually be interpolated bi-parametrically from the bounding curves defining the envelope. To control the whole coordinate system with one single skeleton, however, a proper deformation model must be defined to handle the singularities arising from extreme bending cases. Under this definition, the 'skeleton techniques' described in [15] should not be considered as a skeleton deformation.

Envelope based techniques may be suitable for manipulating a few pictures. When it comes to controlling individual strokes in a complicated drawing, however, it becomes as cumbersome as the use of lines to trace out brush trails. One could see why the place of general brush strokes are not so easily replaceable.

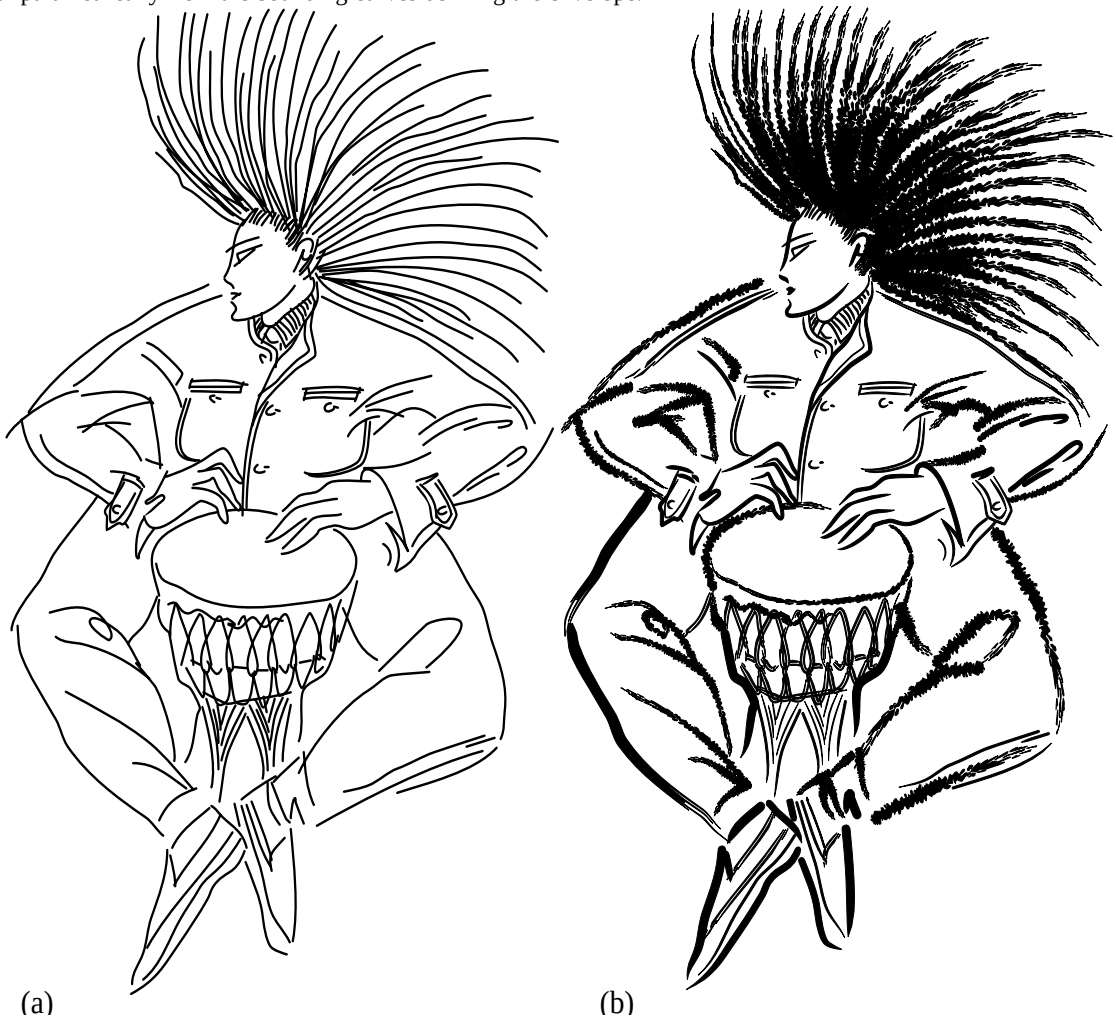

Figure 2: A stylish figure after Tomio Mohri[35]. (a) With a constant thickness stroke. (b) With 5 types of skeletal strokes.

3 EXISTING BRUSH AND STROKE MODELS

3.1 Raster Based Strokes

Many powerful and expressive strokes have been proposed before in SIGGRAPH and other literature. Smith[46] and Whitted[50] proposed the use of antialiased Z-buffer images to create strokes with a 3D appearance. Bleser et. al.[10] used a lookup table of bitmaps indexed by the pressure and tilt reading of a 5D digitizer stylus. They effectively recreated a digital charcoal stick. Another system models graphite pencils in a similar way[49]. Greene's drawing prism[24] even allows the capturing of the path of virtually anything which is being dragged over the prism surface. However, strokes created by these systems suffer from the same problem as those provided by paint programs: after being drawn, they are difficult to edit, especially if they are antialiased.

Other physical models exist. Strassmann[47] modelled the ink-laying processes of bristle brush on paper. Guo and Kunii[25] extended them to include ink-diffusion through the paper fabric mesh. Their results are attractive despite the relatively slow computation speed (1 to 2 minutes per stroke in Strassmann's prototype system). To achieve ultimate authenticity, Pang et.al.[38] even attached real bristle brushes to plotters and defined the strokes by the paths and pen up/down control parameters.

These strokes did reproduce on the digital computer, to a certain extent, the effects made by a brush or pen. Unfortunately, the dexterity demanded of the artist in the handling of the brush or pen is also transferred to the user at the computer, often in an unnatural way. The control parameters (speed of movement, pressure, tilt) of the individual strokes are either dependent on both a powerful input device and a skilful user, or they have to be

specified explicitly. If digital strokes are no less difficult to control than real ones, it makes sense to do the drawings on paper and let the computer do the retouching and other post-processing jobs on the scanned-in version. A further problem is that simulation often takes too long for interactive use on small machines.

Figure 3: A picture drawn using nine different types of skeletal strokes after the Lithograph *The Scream* by Edvard Munch (1863-1944).

3.2 Vector Based Strokes

As for vector graphics based systems, apart from those which depend on constant thickness strokes, there are others following Knuth's line of using analytical paths defined by the trajectory of some geometrical shapes[21,30,31]. The skeleton based stroke primitives in Berkel's SIAS system[9] allowed the local width of a stroke along the path to vary arbitrarily. They might be suitable for specific applications like digital typography. These strokes are however too restrictive in form as a general brush stroke replacement.

In commercial drawing programs, the option to vary the stroke width is already a feature, yet the controlling of it is still far from satisfactory. For systems without a pressure sensitive digitizer, the stroke width is often determined by the drawing speed alone or with other controls[39]. Again, these rather unnatural means of making expressive strokes are difficult to use and time consuming.

3.3 Outcome or the Process?

The monitor screen is not a piece of paper and the digitizer not a pen. Although the brush and stroke metaphor is helpful for understanding the digital drawing process, it may not be wise to strive to model it completely. Since we are usually more interested in the final appearance of the stroke than the physical action of dragging a brush across paper, we might as well try to model the desired look of the stroke in the first place. Of course one could then argue that the mastering of the brush is in itself an art, in which case one should really go back to the use of the physical brush.

In the following sections, we shall first give a brief summary of the key features of skeletal strokes and our drawing program, 'Skeletal Draw'. Then its use as a general drawing tool and its applications in comics and animation making will be explained.

4 SKELETAL STROKES

The use of skeletal strokes[27,28] is a new realization of the brush and stroke metaphor. It does not use physical models (e.g. bristles of brushes or properties of paper) nor use repeated patterns as the basic drawing unit. Instead, an arbitrary picture and its deformation are abstracted into a skeletal stroke to draw with. This structured approach turns out to be far more general than those based on physical models. It is a rich framework for general picture deformation yet the only control parameter required is the brush path itself. The deformation of individual parts of a picture can be independently controlled. Higher order strokes can be used to build up complex pictures, while recursive strokes can be made to generate and manipulate fractals. Some effects that can be achieved include marks made by a bristle brush or a flat nib pen, the effects of wood cut and water-based ink with blotting effects (figure 4). With the use of the higher order anchoring mechanism, it is capable of giving an illusion of objects rotating. This is invaluable in the creation and manipulation of pseudo-3D models.

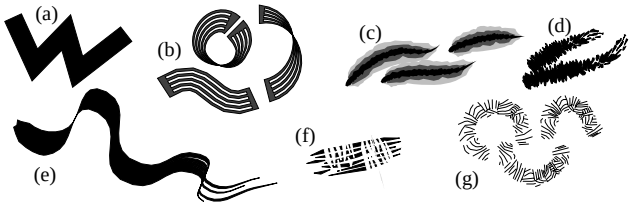

Figure 4: An assortment of effects with skeletal strokes. (a) Constant thickness skeletal stroke, (b) strokes with bending and twisting, (c) water based ink with a blotting effect, (d) 2 applications of a stylish stroke, (e) more than an ordinary flat-nib pen, (f) wood-cut, (g) 3 applications of a stylish stroke.

4.1 Review on Skeletal Stroke Deformation

The skeletal stroke deformation has been discussed in detail in [28], here we will only briefly go through the main points but leave out the mathematical details and derivations.

The skeletal stroke deformation is based on localized parametric coordinate system transformation along the stroke application path. A skeletal stroke is defined by specifying a *reference backbone* and a *reference thickness* on any arbitrary picture. The purpose of defining the reference backbone and thickness is for parameterizing the coordinate system of the original picture. On applying the stroke along a path, the original picture is redrawn on the deformed coordinate system defined by the path, the width of the stroke application, and some other optional parameters like shear angle and twist.

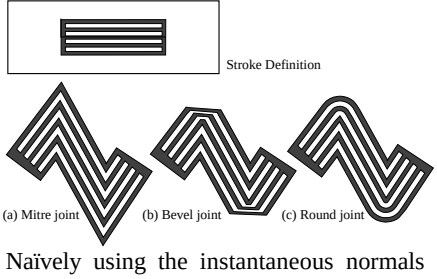

Figure 5: A stroke applied onto the same path with 3 different types of joints.

Naïvely using the instantaneous normals along the application path as the deformed local y axes would cause the flesh (the picture abstracted in a stroke) to wrinkle or fold back onto itself.

To eliminate these undesirable effects, we have proposed a deformation model based on an idealized material with non-localized deformation on axial compression. The lateral deformation is handled by spatially constraining the material to the local centre of curvature. In the cases of extreme bending, which is common for thick polyline strokes, we introduced a *macroscopic centre of curvature* (MCC) which is the intersection point of two angle bisectors at two adjacent joints. The coordinate space would converge towards the MCC as in the case for continuous stroke paths. The joint zone coordinate systems for mitre, bevel and round joints are also derived.

If we use an undistorted skeletal stroke as a texture coordinate space[11], sampled images can be deformed with a skeletal stroke application using standard texture mapping techniques. Since the relation between the original (defined by the reference backbone) and the distorted (specified in the stroke application) coordinate systems is well defined, proper filtering of the texture map can be performed.

Normally, the flesh would freely stretch and shrink with the length of the application path. This might not be desirable for special features in a stroke, like serifs or other stroke-end forms. We have therefore defined an anchoring mechanism which allows the aspect ratio of part of the flesh (i.e. a subset of vertices or control points in the original picture) to be retained. This is actually achieved by parameterizing the coordinates to a new coordinate system with a shifted origin known as the *anchor origin*. The final coordinates (both x and y) on application are controlled by the width of the stroke application alone, hence a part of the flesh anchored to a single anchor origin would retain its aspect ratio and would not stretch or shrink with the application path length but would move with the final position of the anchor origin. If we anchor different parts of the flesh to different anchor origins, they would shift to different absolute positions.

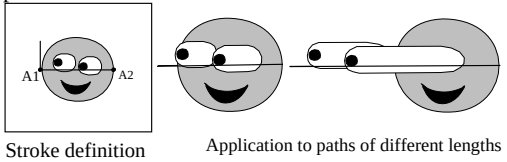

Figure 6: Anchoring different parts to different anchor origins.

The effect of anchoring to different anchor origins can be interesting. In figure 6, the figure's pupils (and half of each eye) and the rest of its face are both anchored but to different ends of the stroke. The former is anchored to the point A1; the latter to A2. If we change the stroke length while fixing the stroke width, the eyes will bulge with the end of the stroke while the rest of the face stays with the beginning of the stroke.

If the anchor origins themselves are anchored to other anchor origins, the deformation result would be even more dramatic. In fact, we have proved that for *n*th order anchoring, the final x and y coordinates are actually *n*th degree parametric equations with the coefficients being the respective x and y coordinates of the various anchor origins and the parameter being the changes in aspect ratio from the origin aspect ratio on definition[28]. Therefore we could conveniently encode any subtle deformations to a single stroke definition.

The elegance of the general anchoring mechanism lies in its consistency with the intuitive zeroth order anchoring and arbitrary deformation. The extension to piecewise continuous general anchoring and the use of this mechanism to construct pseudo-3D models shall be described in a subsequent section.

Figure 7: A stroke resembling a head. 2nd order anchoring has been used to control the deformation of the head to give an illusion of rotation. The anchor origins are determined automatically by the system.

5 SKELETAL DRAW: A SKELETAL STROKE BASED DRAWING SYSTEM

Skeletal Draw is an implementation of our vision of a professional drawing system (Figure 21). The current system is implemented on a 486 PC under MS-Windows™, and is derived from a previous Unix™ version on X.

The system consists of less than 15000 lines of C++ code. Basic drawing tools like the free-hand brush, the polygonal pen, rubber rectangles and so on are all available; their uses are basically identical to those in most other drawing systems. The major difference is that the strokes are directional and much more expressive than the usual constant thickness strokes.

5.1 Drawing with Skeletal Strokes

Drawing with skeletal strokes is similar to drawing with constant thickness strokes on other systems. To select a stroke to be used for a particular path, there is a repertoire of predefined or user-defined strokes to choose from in the menu. However, choosing from the menu is inconvenient, especially when artists are likely to use a variety of strokes interchangeably. To ensure quick switching between strokes, we therefore allow the binding of 40 different strokes to the numeric keys (together with a combination of the Shift and Control modifier keys). The system can of course take advantage of more powerful input devices like the pressure sensitive stylus. Skilful artists may want to modify the stroke attributes and to switch between strokes based on the pen pressure and speed. It is however worth pointing out that all the illustrations shown in this paper were created with just a modest 2-button stylus.

Besides varying the application path, a stroke application can be modified by specifying the stroke width and shear angle. This is done by interactively pulling out a rubber line from any one end of the stroke (Figure 21) or by specifying explicit numeric values. The length of the line indicates the stroke width and the angle the line makes with the path's normal would give the stroke the same amount of shearing. The direction of application can also be reversed by hitting a key. The colour and fill pattern of a stroke can also be changed without touching the objects in the original definition. Figures 20 and 22 are drawn using the same set of strokes used in Figure 2 and 3. After modifying an applied stroke, the system will immediately redraw on the screen the stroke resulting from the new application.

As is the case with PostScript[2], the stroking process (the application of a skeletal stroke along a path) is separated from the filling process. An area enclosed by a skeletal stroke path can be filled in the same way as polygons enclosed by constant thickness strokes are.

5.2 Defining New Strokes

Defining a new stroke is easy. The user simply drags out a stroke definition box in the middle of which is an L-shaped line

(Figure 8). The horizontal part and the vertical part of the L-shaped line represents the stroke's reference backbone and the reference thickness respectively. The L-shaped line provides a scale for the new stroke, and everything within the box would be included as part of the stroke. If the orientation of the new design is not aligned with the reference backbone, the user can always transform the box (rotate, translate, scale or even put it in perspective) so that the L-shaped line is aligned with the picture before committing the definition. The inverse of the alignment transformation of the box will be used to transform the coordinates of the vertices or control points of the picture before parametric coordinates of the new stroke are recorded.

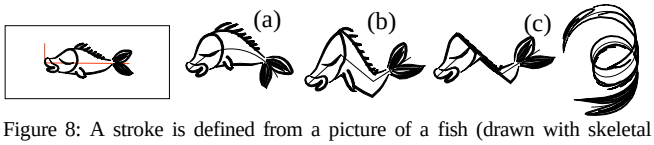

Figure 8: A stroke is defined from a picture of a fish (drawn with skeletal strokes). The stroke is applied (a) on a curved path, (b) on a polyline path, (c) using the ribbon style.

5.3 Higher Order Strokes

A stroke can be defined to consist of other skeletal strokes, in which case the stroke defined would be a higher order stroke. Actually, a stroke can even be defined in terms of itself by first laying out the applications of a special stroke named 'self'. Since the appearance of the stroke 'self' is insignificant, it is normally defined as a constant thickness rectangular stroke. It needs to be there for the user to specify the relation between the recursive applications and the entire stroke itself.

Deferred Application or Not

In the definition of a higher order stroke, we are faced with the question of whether to treat the appearance of the sub-strokes as geometric attributes which deform with the main stroke or just cosmetic attributes akin to pseudo-pen size in the final coordinate space[12]. The answer to this determines whether we apply the sub-strokes before defining the main stroke (sub-strokes being merely geometric objects of the main stroke) or whether we apply them after applying the main stroke (i.e. sub-strokes are applied onto newly deformed application paths). The former option would not involve application of the sub-strokes to potentially curlier paths and are referred to as the *flattened definition*. Flattened definition is easy to handle and is more efficient. It would, however, fail to take advantage of an important feature provided by the skeletal stroke framework: the anchoring mechanism. Furthermore, it cannot be used with self recurring strokes, since it is not possible to determine the 'self' stroke's appearance before it has been applied. In this case, either the *deferred application* (the second option) or the Measured Rendering Algorithm[4,5,6] must be adopted.

In our system, both options are available with deferred application as the default. With the case of deferred application, we still have to decide what to do with the resulting width of the sub-strokes. Under this mode, the resulting widths at different parts of a sub-stroke would differ in relation to the position and orientation of the main stroke. Even for strokes undergoing affine transformations, the determination of the 'correct' width is a problem. For sub-strokes subjected to highly irregular skeletal transformation, this becomes very difficult. Possible approaches adopted by us are to either reduce the width of sub-strokes to an invariant or to scale it in proportion to the width the resulting main stroke. In general, the results based on the latter (which is the adopted option) are found to be satisfactory. In cases where they are not, we can always go back to a flattened definition.

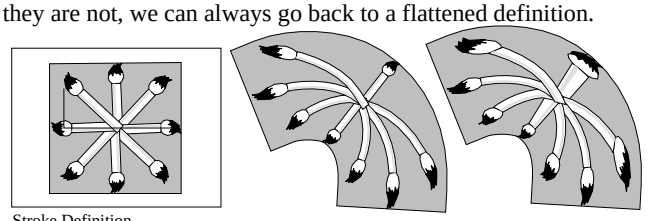

Figure 9: The stroke definition on the left is made up of a square and 8 applications of a stroke resembling a brush. Different results with deferred application (middle) and flattened definition (right) are shown.

Rendering a Recursive Stroke

Rendering a recursive stroke explicitly by determining the resultant geometry is in general impossible. Since a recursive stroke is equivalent to an IFS (Iterated Function System) code (with a condensation set if objects other than itself are present in the definition)[3,4,5,6], the attractor is often a fractal with a fractal dimension less than 2. We therefore have to resort to one of the approximation methods for rendering these fractals. The Escape Time Algorithm[4] would result in a clean and sharp image while the Measured Rendering Algorithm[4,5,6] would give a quick but rough image of the attractor. Here, the decision on which algorithm and level of accuracy to use is left to the discretion of the user. Some sample recursive strokes are shown in Figure 9. One interesting point that is worth a small remark if the Escape Time Algorithm is used: after recurring to a certain stage, something must be drawn out by the system. In our system, this something can be specified by the user to be any arbitrary skeletal stroke. This technique is particularly handy for drawing plants and scattered objects. Furthermore, we could waive the contractive requirement of the transformations (the sub-strokes, in our case) as we are using the IFS to draw pictures but not to encode pictures.

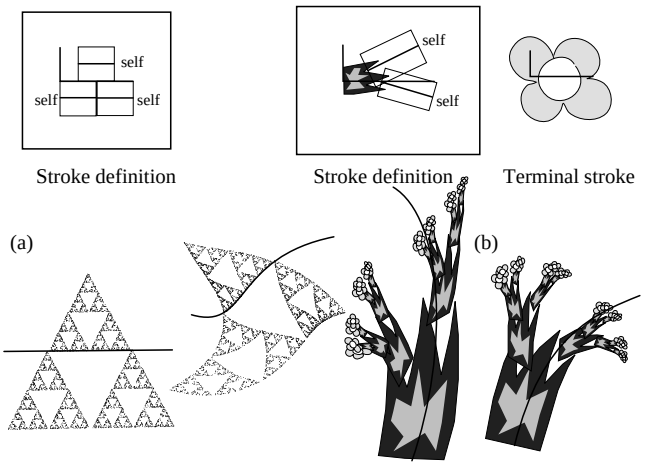

Figure 10: Recursive stroke. (a) A pure recursive stroke defined in terms of 3 instances of itself. On application, it gives a Sierpinski triangle. The stroke can be deformed just like any other stroke. (b) A recursive stroke defined in terms of 2 instances of itself and 2 polygons. The stroke application has recurred to 5 level deep and a terminal stroke of a flower is drawn at the last level. (The flower has been anchored to retain its aspect ratio).

5.4 Anchoring the Flesh

Zeroth order anchoring is the more often used anchoring mode. It preserves the aspect ratio of a chosen part of a picture so that after

application that part would not stretch or shrink with the length of the application path. To perform a zeroth order anchoring is easy. In the stroke definition box, one just needs to use the Anchor Tool to circle the part of the flesh to be anchored and to specify an anchor origin, the position (within the box) to where that part of the flesh is to be anchored. All the vertices and control points inside the circled region would then be anchored to the anchor origin accordingly.

For higher order anchoring, it is not at all easy to figure out where to put the higher order anchor origins to achieve a particular effect. Instead of leaving the user to do this by trial and error, our solution to this is specification by examples. The user supplies an arbitrary number of instances of the picture, each of which corresponds to a different factor of the aspect ratio of the original picture. The system can then calculate the corresponding anchor origins.

Anchoring Sub-Stroke Width and Shear Angle

If the given picture instances contain sub-strokes and the width and shear angle of the sub-strokes are not identical in different instances, then we must also find a way to describe these variations. The solution we have adopted is to record the width and shear angle of a stroke in the form of an additional point. The angle of the line joining this special *shear-width point* to the end of the stroke and the tangent of the end would be the shear angle and the length of this line the stroke width. This is identical to our use of a rubber line to interactively specify the width and the shear angle. This special point is then anchored in exactly the same way as other points.

Piecewise Continuous Anchoring

If N instances are given, it is theoretically possible to use N-1 anchor origins to define a deformation that would result in the N given forms at the corresponding aspect ratio factors. Just as in curve fitting, we would not fit a (n-1)th degree polynomial to a set of n points because of the over-crookedness of the resultant curve and the lack of localized control over the curve. Instead of determining the multi-level anchor origins which are the coefficients of the two continuous parametric equations, we could also employ piecewise continuous anchoring. In this case, the anchor origins (coefficients of the piecewise continuous parametric equations) can be easily determined from the neighbourhood picture instances. In the Skeletal Draw system, a Catmull-Rom spline interpolation scheme is employed.

Figure 11: Two stylish cartoon figures drawn with skeletal strokes.

5.5 Information Recorded in a Stroke

The amount of information that needs to be stored for a skeletal stroke is small. For a stroke definition it is merely the original picture plus the original aspect ratio of the stroke (the ratio of length of reference backbone to reference thickness). If the points are anchored, the list would include the order of anchoring and the corresponding anchor origins.

For a stroke application, only the application path, its width (and other optional parameters) and its reference to a stroke definition is recorded. If the definition of a stroke is modified, the appearance of all the stroke applications referring to that stroke would reflect the changes immediately on redraw.

6 General Drawing Applications

We anticipate that skeletal strokes would have considerable impact on many drawing applications: in fashion design where stylish bold strokes are often used; in interior design where pseudo-3D strokes representing furniture and architectural objects can be laid out with ease; in dynamic clip-arts which can deform dramatically beyond the limits of stretching, shrinking, shearing, twisting and bending. If implemented in windowing system kernels or page description languages like PostScript, the skeletal stroke deformation algorithm would extend the systems to permit expressive strokes to be specified in more or less the same way as the constant thickness strokes currently are, think about windows with borders in stylish strokes. In the case of PostScript, it would be something like:

```
/name setstroke      % select a skeletal stroke
newpath ..moveto
..lineto ..curveto   % path construction
5 setstrokewidth     % optional controls
stroke               % stroke application
```

Figure 12: Mimicking Chinese brush painting.

7 Comics and Animation

One interesting application of skeletal strokes that we have been exploring is its use in comics and stylish 2½D animation production. Snoopy and the characters in 'The Yellow Submarine' are characters falling under this category[18]. Comics is a now a highly respected sub-culture in many countries especially in Japan. Works by famous comics artists like Tezuka Osamu[37] have even been studied and given literary praise. Pragmatically, comics publishing is also a mega-dollar business. Even though

demand for quality comics is high, the production of comics is still mainly a hand craft. Day in and day out, teams of overworked assistants are painstakingly filling in the buildings and pedestrians in the background and drawing in special dramatic effects to meet deadlines. Computers have not yet been able to offer substantial help because of the lack of efficient support for stylish strokes, which are essential for creating variety and atmosphere creating, and hence important to the success of a comics work.

Skeletal strokes with the use of the general anchoring mechanism could provide the comics artist with a library of faces, limbs and features which are pseudo-rotatable (Figure 7). Colour gradation and shading changes of the models can also be encoded into the strokes using gradation lines or points[14]. This library could be created by the artist at the character design process. With it the artist could quickly lay out the characters in various postures and positions. Wildly different looks that are not readily available from the library can be introduced by editing the looks of applied strokes. Without sacrificing the flexibility of hand drawing, this approach not only speeds up the production process dramatically, but also helps to make sure characters have a consistent form. The time consuming task of drawing buildings and other objects in the background can also be sped up with a library of pseudo-3D architectural strokes and recursive plant strokes.

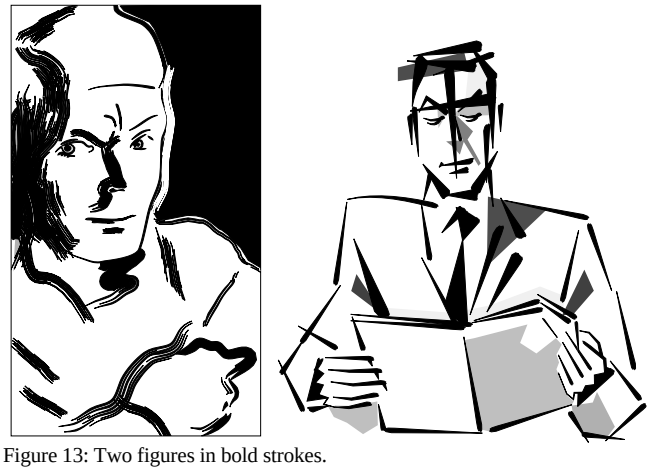

Figure 13: Two figures in bold strokes.

7.1 Speed Lines for Motion Blurring

In both comics and animation, fast action is often represented by speed lines, which suggest the distance an object or character has travelled across the camera before its shutter is closed again. Motion blurring is particularly important in animation because it reduces temporal aliasing effects. Its method of generation in vector based cartoon has not been, however, a well addressed problem.

One possible way is to employ temporal antialiasing techniques like stochastic sampling across the time dimension[19,20] or the recently proposed more efficient technique of Spatio-temporal Filtering[45]. These methods involve considerable computational cost which would not be practical for interactive use unless on high-end machines. 2½D image based techniques have also been proposed[33] which are more efficient. However all these techniques require the resultant picture to be a raster image and therefore might not always be appropriate for use in an interactive vector-based drawing system. In traditional comics, the use of crisp lines and strokes to indicate motion is usually the preferred way. We have therefore implemented a simple method to efficiently generate speed lines based on the traditional approach for blurring cartoon objects.

The traditional technique used by famous animators like Shamus Culhane is to do nothing to the leading edge of the moving object but blur the trailing edge with a trail (which is the same colour as the object) along the direction of motion[18]. Based on this approach, our system generates speed lines as follows. The moving object is a polygon enclosed possibly by skeletal strokes (the polygons inside a skeletal stroke are handled in the same way). First we have to determine which points outlining the polygon make up the trailing edge of the object. For each evenly spaced outlining point then, this question is asked: does a point a very small distance away from it along the direction of the velocity lie inside or outside of the polygon? If it lies inside, the outlining point is on the leading edge, and nothing would be done to it. For points on the trailing edge, each would be attached with a thin triangle the colour of the polygon. The base of the triangle would coincide with the outlining point; the apex would lie at a distance proportional to the speed along the velocity direction.

The spacing of the speed lines now varies with the curvature of the trailing edge, or, to be precise, the sine of the angle θ between the velocity direction and the tangent of the polygon at the base of the speed line. This variation in spacing is annoying especially at curved trailing edges. To achieve uniform spacing between the resultant speed lines, we have to correct the step size of the speed line base point along the polygon by the factor $\sin(\theta)$. We only draw the next speed line when the accumulated corrected spacing is greater than the specified spacing (Figure 14).

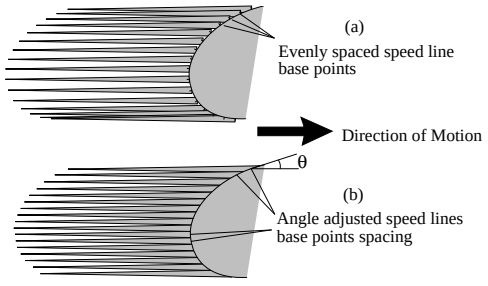

Figure 14: Adjusting the speed line spacing.

Figure 15: A keyframe drawing for the animation sequence, *The Cat in Skeletal Strokes*. The speed lines in this still picture are generated by manually moving the umbrella and the hat. The ones in the animation are derived automatically from object movements.

To introduce more variety to speed lines, which is particularly important for static pictures, the spacing and length of the speed lines are given a certain randomness (Figure 15). In fact, there is no reason why speed lines cannot be shapes other than

triangles. Why not arbitrary skeletal strokes along curved speed lines? So this is what we have done. To give an object drawn with Skeletal Draw a motion blur, the user only needs to transform the object to a new configuration (position and orientation). Using the specified stroke and speed line spacing, the system would create the speed lines from the new configuration to the old one. If a few intermediate configurations are marked before generating the speed lines, their paths would then be Catmull-Rom splines interpolating the various configurations.

7.2 Controlling a Pseudo-3D Stroke

We have demonstrated how to create an illusion of rotation by making use of the general anchoring mechanism. To perform that deformation, only one control parameter, the aspect ratio of the stroke, has been used. We could in theory control any kind of deformation of a picture with the aspect ratio alone. If different features in a picture are to change in different ways, however, it would be more convenient to have independent controls.

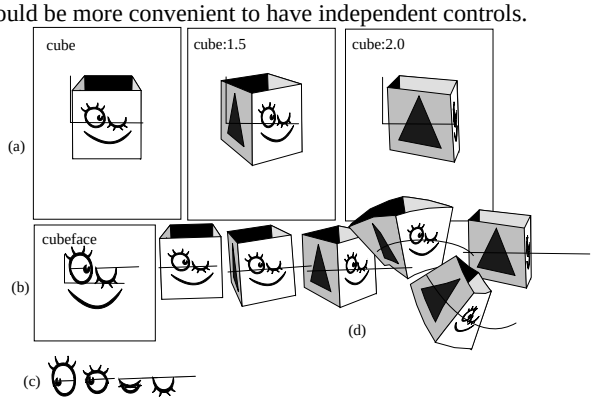

Figure 16: Hierarchical definition allowing independent control of different parts. (a) The definition of the stroke 'cube', 2nd order anchored by providing 3 instances at 1, 1.5 and 2 times its original aspect ratio(i.e. the left most one). (b) The definition of the sub-stroke 'cubeface' used in the definition of the stroke 'cube'. Notice that the eyes in the 'cubeface' stroke are also anchored skeletal strokes. The right eye has been closed by lengthening the application path. (c) Demonstrating how the 'eye' closes on varying its stroke length. (d) Application of the stroke 'cube' at different path lengths. Notice that the closing of the right eye takes effect at any 'angle' of the cube stroke.

We shall explain how to introduce extra independent controls with an example. Figure 16 shows a stroke of a cube which has a face on one of its sides. We can independently control the rotation of the cube and the blinking of the eyes of the face. This is done by defining the cube as a main stroke with the face as its sub-stroke. As was mentioned in Section 5.5, each stroke application is recorded with reference to the stroke definition (or stroke definitions if sub-strokes exist). This means that any changes in the definition of the stroke 'cubeface' would propagate to that of the stroke 'cube' and the changes would be reflected in its resulting stroke applications. Now using general anchoring, we can define the eyes on the stroke 'cubeface' to blink by varying the stroke application length of the 'eye' stroke. Hence by manipulating the lengths of the definitions of the strokes 'cube' and 'eye', we can control the cube rotation and eye blinking actions independently. This feature is particularly attractive for animating stylish cartoon characters which have no actual 3D realization.

This hierarchical way to define skeletal strokes means it is possible to build up powerful 2½D models with an arbitrary number of independent control parameters. Actually, these controls can be raised from the stroke definition level: we can easily set up an external control system by which users can manipulate different features with pop-up sliders, dials or other graphical user interfaces. We shall not go into details here.

Figure 17: A picture after a CACM front cover[1].

7.3 Animation with Skeletal Draw

Animation takes full advantage of the picture and deformation abstraction capability provided by skeletal strokes. The Skeletal Draw program has also been built with facilities to generate animation sequences. Traditional key-frame technique is used by Skeletal Draw. To create the key-frames, the user directly manipulates the characters in the scene. The system then calculates the in-betweens by interpolating Catmull-Rom splines. After the keyframes have been laid out, the program can do a quick playback using fewer in-betweens and substituting any skeletal strokes with rectangular (possibly deformed) ones. This quick preview is mainly to let the user to have a feel for the timing and smoothness of the motion. The in-betweens can be recorded in the same formats as the key-frames for inspection or further refinement. Because we are interpolating only the skeletons and they are usually of very simple forms, ensuring a proper interpolation of them is trivial. Furthermore, the shapes of the flesh are totally controlled by the skeletons, no interpolation of the individual points on the flesh is performed. Therefore the ambiguity of the shape blending problem[41,42] on the flesh would not arise at all.

The use of general anchoring is not limited to character creation. Since an arbitrary deformation can be recorded in one single stroke, a skeletal stroke can be viewed as an encapsulated unit of motion. These 'canned' motions can be applied conveniently at any time in a sequence. By redefining the sub-strokes, these motions can be used to animate other objects.

Stylized and complex drawings like those by Maurice Sendak[43] and Dr. Seuss[44] are known to be difficult to animate. It took Oscar winning animator Gene Deitch five years of experimenting before he found a way to transfer Maurice Sendak's drawing techniques to the screen[18]. With skeletal strokes' compact abstraction, complicated hatching or stipples can be condensed into simple units with which to further build up characters.

We have created a 1 minute animation sequence, *The Cat in Skeletal Strokes*, based on Dr. Seuss' *The Cat in the Hat*[44] character using the Skeletal Draw program. Two man days were used to create the sequence, which could have been even shorter

had the bugs which appeared then been fixed earlier. Taking into consideration that our system was run on a 486 PC, this efficiency compares even more favourably than the Inkwell system[32]. The most time consuming part in the course of production is actually in the generation of antialiased frames.

Figure 18: A rasterized frame from *The Cat in Skeletal Strokes*. The length and direction of the speed lines changes automatically with object movements.

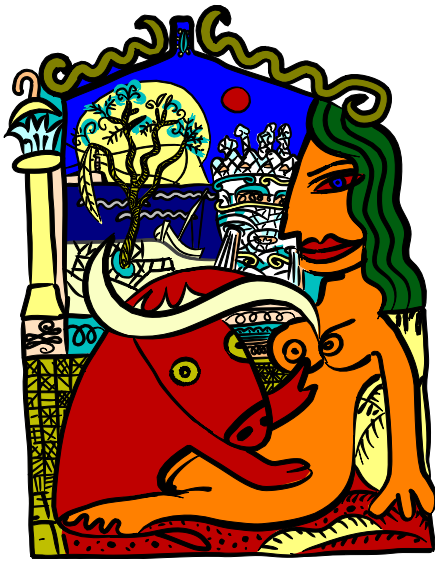

Figure 19: A picture after Javier Mariscal's *Spain, the New Rising Star* (1990). A few strokes have been specially defined for this picture, e.g. the olive-green wiggly deco on the roof top and the hotel in the backdrop.

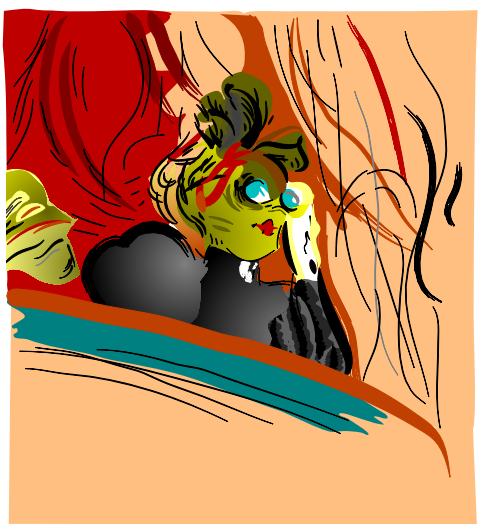

Figure 20: After the colour-lithograph *Loge with Golden Mask* (Detail) by Henri de Toulouse-Lautrec (1864-1901).

7.4 Incorporating Other Techniques

Metamorphosis or Morphing techniques[8,53] have received great attention lately due to their capability to generate dramatic but convincing deformations to photographic images with relatively simple calculations. By defining skeletal strokes to consist of morphing control lines on sampled images, we could build up photorealistic pseudo-3D models of images instead of vector based pictures. With the hierarchical control over various parts of the strokes, we could start to make Michael Jackson's eyes blink and his head turn independently instead of our cartoon character's.

Figure 21: The width and shear angle of the yellow 'sausage' stroke (detail of the picture in Figure 19) is being modified.

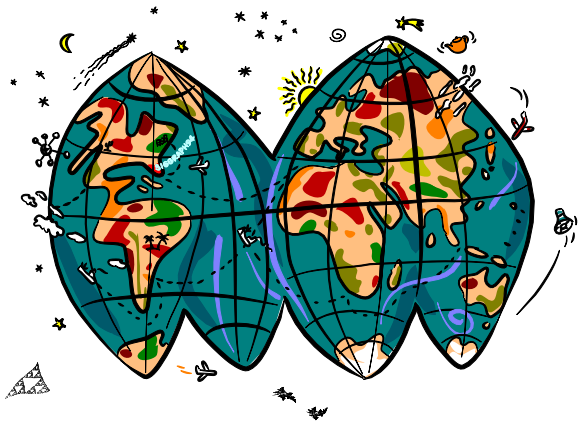

Figure 22: *Where on Earth is SIGGRAPH94*. After a 'Cobi' poster by Javier Mariscal. Notice the fractal strokes at the lower part of the picture.

There are also published techniques which are useful for high level specification of animation sequences. For example, skeleton based 3D animation techniques based on physical deformations of complex 3D models[16,23]. While these may be useful for realistic deformations, they are not quite applicable to stylish figures, which do not even have a 3D realizable form. Furthermore, physical simulation is often unnecessary in stylish animation, which is the field we have been focusing on.

Techniques like gait control[22], constraint based methods [7,29,36,40,51,52] and dynamics simulation[13,46] are certainly important for creating highly realistic and complex physically based motions. Incorporation of these methods into our system is worth exploring in future.

9 CONCLUSION

We have presented in this paper the skeletal strokes framework for general drawing and animation. The framework has been demonstrated with the skeletal stroke based drawing system, Skeletal Draw. The decisions involved in its implementation and

the resultant interaction style have been presented. In fact all the pictures and illustrations in this paper are drawn by the authors with the system without the help of a scanner. Figures 2, 3, 12, 19-22 in particular, demonstrate that skeletal strokes can be used to create rich and complex drawings from scratch. In the past, these can only be practically drawn with paint programs.

We have also introduced the combined usage of the hierarchical higher level stroke definition and the general anchoring mechanism to create pseudo-3D models with an arbitrary number of independent controls over different features of the model. The significance of these aspects in sample applications like comics and 2½D animation has also been discussed.

REFERENCES

[1] Adams, Kathryn. [picture] In *Communications of the ACM*, October 1993, front cover.

[2] Adobe Systems Inc. *PostScript Language: Reference Manual*. Addison-Wesley, 1989.

[3] Barnsley, Michael F. *Fractals Everywhere*. Academic Press, 1988.

[4] Barnsley, Michael F. and L. P. Hurd. *Fractal Image Compression*. AK Peters Ltd., 1993.

[5] Barnsley, Michael F., A. Jacquin, F. Malassenet, L. Reuter and A. D. Sloan. Harnessing Chaos for Image Synthesis. Proceedings of SIGGRAPH88. In *Computer Graphics*, 22, 4 (August 1988).

[6] Barnsley, Michael F. and A. D. Sloan. A Better Way to Compress Images. In *BYTE*, January 1988.

[7] Barzel, Ronen and A. H. Barr. A Modelling System Based on Dynamic Constraints. Proceedings of SIGGRAPH88. In *Computer Graphics*, 22, 4 (August 1988).

[8] Beier, Thaddeus and S. Neely. Feature-Based Image Metamorphosis. Proceedings of SIGGRAPH92. In *Computer Graphics*,26,2 (July 1992).

[9] van Berkel, Pierre. SIAS, Strokes Interpreted Animated Sequences. In *Computer Graphics Forum*, 8, 1989.

[10] Bleser, Teresa W., J. L. Sibert and J. P. McGee. Charcoal Sketching: Returning Control to the Artist. In *ACM Transactions on Graphics*, 7(1):76-81, January 1988.

[11] Blinn, J. F. and M. E. Newell. Texture and Reflection in Computer Generated Images. In *Communications of the ACM*, volume 19, Oct., 1976.

[12] Bresenham, J. E. Ambiguities in Incremental Line Rastering. In *IEEE Computer Graphics and Applications*, volume 7, 1987.

[13] Brunderlin, Armin and T. W. Calvert. Goal-Directed, Dynamic Animation of Human Walking. Proceedings of SIGGRAPH89. In *Computer Graphics*, 23, 3 (July 1989).

[14] Burtnyk, N. and M. Wein. Computer-Generated Key-Frame Animation. In Journal of Society for Motion Picture and Television Engineers, 80:149-153, 1971.

[15] Burtnyk, N. and M. Wein. Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. In *Communications of the ACM*, volume 19, October 1976.

[16] Chadwick, John E., D. R. Hauman and R. E. Parent. Layered Construction for Deformable Animated Characters. Proceedings of SIGGRAPH89. In *Computer Graphics*, 23, 3 (July 1989).

[17] Chen, Shenchang E. and W. Lance. View Interpolation for Image Synthesis. Proceedings of SIGGRAPH93. In *Computer Graphics*, Annual Conference Series, 1993.

[18] Culhane, Shamus. *Animation from Script to Screen*. Columbus Books, 1989.

[19] Cook, Robert L. Stochastic Sampling in Computer Graphics. In *ACM Transactions on Graphics*, 5(1), January 1986.

[20] Dippé, M. A. Anti-aliasing Through Stochastic Sampling. Proceedings of SIGGRAPH85. In *Computer Graphics*, 19, 3 (July 1985).

[21] Ghosh, Pijush K. and S. P. Mudur. The Brush-Trajectory Approach to Figures Specification: Some Algebraic-Solutions. In *ACM Transactions on Graphics*, 3(2):110-134, April 1984.

[22] Girard, Michael and A. A. Maciejewski. Computational Modelling for the Computer Animation of Legged Figures. Proceedings of SIGGRAPH85. In *Computer Graphics*, 19, 3 (July 1985).

[23] Gourret, Jean-Paul, N. Magnenat Thalmann, D. Thalmann. Simulation of Object and Human Skin Deformation in a Grasping Task. Proceedings of SIGGRAPH89. In *Computer Graphics*, 23,3 (July 1989).

[24] Greene, Richard. The Drawing Prism: A Versatile Graphic Input Device. Proceedings of SIGGRAPH85. In *Computer Graphics*, 19, 3 (July 1985).

[25] Guo, Qinglian and T. L. Kunii. Modelling the Diffuse Paintings of 'Sumie'. In *Modelling in Computer Graphics*. T. L. Kunii, Ed. Springer Verlag. 1991.

[26] Hahn, James K. Realistic Animation of Rigid Bodies. Proceedings of SIGGRAPH88. In *Computer Graphics*, 22, 4 (Aug. 1988).

[27] Hsu, Siu Chi. *Computer Support for Large Character Set Languages*. PhD thesis, University of Cambridge, Computer Laboratory, December 1991.

[28] Hsu, Siu Chi, I. H. H. Lee and N. E. Wiseman. Skeletal Strokes. In *UIST'93 Proceedings of the ACM SIGGRAPH and SIGCHI Symposium on User Interface Software and Technology*, November 1993.

[29] Issacs, Paul M. and M. F. Cohen. Controlling Dynamic Simulation with Kinematic Constraints, Behaviour Functions and Inverse Dynamics. Proceedings of SIGGRAPH87. In *Computer Graphics*,21,4 (July 1987).

[30] Knuth, Donald E. Lessons Learned from METAFONT. In *Visible Language*, 19(1), 1985.

[31] Knuth, Donald E. *The METAFONT Book*. Addison Wesley, 1989.

[32] Litwinowicz, Peter C. Inkwell: A 2½D Animation System. Proceedings of SIGGRAPH91. In *Computer Graphics*, 25, 4 (July 1991).

[33] Max, Nelson L. and D. M. Lerner. A Two-and-a-Half-D Motion-Blur Algorithm. Proceedings of SIGGRAPH85. In *Computer Graphics*, 19, 3 (July 1985).

[34] Miles, Linda and B. Wilson. *Illustration Techniques with Adobe Illustrator for Windows*. Hayden (a Prentice Hall division), 1992.

[35] Mohri, Tomio. Issey Miyaki Advertisement. [picture] In *Joyce Men Spring 1993*. Joyce Publishing Ltd., 1993.

[36] Ngo J. Thomas and J. Marks. Spacetime Constraints Revisited. Proceedings of SIGGRAPH93. In *Computer Graphics*, 1993.

[37] Osamu, Tezuka. Hinotori (Fire Bird). Kadokana Bunko, 1990.

[38] Pang, Y. J. and H. X. Zhong. Drawing Chinese Traditional Painting by Computer. In *Modelling in Computer Graphics*. T. L. Kunii, Ed. Springer Verlag. 1991.

[39] Paulson, Ed. *Using CorelDraw! 4*. Que Corporation, 1993.

[40] Platt, John C. and A. H. Barr. Constraint Methods for Flexible Models. Proceedings of SIGGRAPH88. In *Computer Graphics*,22,4 (Aug 1988).

[41] Sederberg, Thomas W. and E. Greenwood. A Physically Based Approach to 2D Shape Blending. Proceedings of SIGGRAPH92. In *Computer Graphics*, 26, 2 (July 1992).

[42] Sederberg, Thomas W., P. Gao, G. Wang and H. Mu. 2D Shape Blending: An Intrinsic Solution to the Vertex Path Problem. Proceedings of SIGGRAPH93. In *Computer Graphics*, 1993.

[43] Sendak, Maurice. *Where the Wild Things are*. HarperCollins Publishers Ltd., 1992.

[44] Dr. Seuss. *The Cat in the Hat*. Random House, 1958.

[45] Shinya, Mikio. Spatial Anti-aliasing for Animation Sequences with Spatio-temporal Filtering. Proceedings of SIGGRAPH93. In *Computer Graphics*, 1993.

[46] Smith, Alvy Ray. Paint. In *Tutorial: Computer Graphics*, 13(2):501--515. IEEE Press, 1982.

[47] Strassmann, Steve. Hairy Brushes. Proceedings of SIGGRAPH86. In *Computer Graphics*, 20, 4 (Aug. 1986).

[48] Terzopoulos, Demetri and K. Fleischer. Modelling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. Proceedings of SIGGRAPH88. In *Computer Graphics*, 22, 4 (Aug. 1988).

[49] Vermeulen, Allan H. and P. P. Tanner. PencilSketch -- A Pencil-Based Paint System. In *Proceedings of Graphics Interface '89*, May 1989.

[50] Whitted, Turner. Anti-Aliased Line Drawing Using Brush Extrusion. Proceedings of SIGGRAPH83. In *Computer Graphics*,17,3 (July 1983).

[51] Witkin, Andrew, K. Fleischer and A. Barr. Energy Constraints On Parameterized Models. Proceedings of SIGGRAPH87. In *Computer Graphics*, 21, 4 (July 1987).

[52] Witkin, Andrew and M. Kass. Spacetime Constraints. Proceedings of SIGGRAPH88. In *Computer Graphics*, 22, 4 (Aug. 1988).

[53] Wolberg, George. *Digital Image Warping*. IEEE Computer Society Press, 1990.