Computerising 2D Animation and the Cleanup Power of Snakes.

Fionnuala Johnson

Submitted for the degree of Master of Science
University of Glasgow,
The Department of Computing Science.

January 1998

Abstract

Traditional 2D animation remains largely a hand drawn process. Computer-assisted animation systems do exists. Unfortunately the overheads these systems incur have prevented them from being introduced into the traditional studio. One such problem area involves the transferral of the animator's line drawings into the computer system. The systems, which are presently available, require the images to be over-cleaned prior to scanning. The resulting raster images are of unacceptable quality. Therefore the question this thesis examines is; given a sketchy raster image is it possible to extract a cleaned-up vector image? Current solutions fail to extract the true line from the sketch because they possess no knowledge of the problem area. However, snakes use prior knowledge about the nature of sketchy images to determine the correct line from several possibilities. As a snake is an energy minimising spline, the result is in vector format. Therefore in extracting the clean line from the sketch the conversion from raster to vector is also achieved. This technique makes snakes less prone to errors than the current algorithms. In reducing the errors, the overhead produced when transferring the drawings from paper to computer is also reduced.

Contents

List of Figures

Acknowledgments

I would like to thank my supervisor, John Patterson, University of Glasgow, for his help, advice and encouragement throughout the research that is described in this thesis. I would also like to thank Ray Welland, University of Glasgow for providing valuable feedback on early drafts of this document.

Chapter 1

Introduction:

1.1 Motivation behind this Thesis.

> Animation, at best, is a costly procedure, in both time and money, and anything that eases its birth process should not be ignored. If audiences only knew all that is involved in any animated production, their respect for what I consider one of the most creative art forms would increase. [20]

With today's advances in computer graphics, it is incredible to think the majority of traditional 2D animation is still done by hand. In fact the process has changed little over the last 50 years. Yet other areas of computer graphics have blossomed, as can be seen in the many spectacular 3D commercials and art works. Why has the expensive time-consuming process of 2D animation been left behind?

It would be untrue to think that computers are never used in the traditional 2D studio. The Disney studio have been employing computers on an increasing basis over the last decade [4]. One area where computers have successfully been introduced is in the creation of complicated background scenery. A classic example is the ballroom scene in Beauty and the Beast. A 3D model of the ballroom was created and rendered with elaborate marble effects. The whole room was lit with a multitude of candles. As the couple danced around, the camera view of the room also rotated. Maintaining continuity in the lighting effects of the flickering candles

on the ever changing marble would have been a difficult task for the traditional animator. But as each effect could be modelled, all the necessary calculations could now be carried out by the computer. The result is spectacular.

Building 3D models is a very time-consuming task, and so the backgrounds must prove sufficiently difficult to merit resorting to 3D modelling. However, the nature of backgrounds make them ideal for 3D technology, they convey the illusion of depth and remain unchanged for long sections of the film. The same is not true for traditional 2D animation, whose appearance is flatter than its computerised 3D equivalent. Comparing Disney's purely 3D animated film "Toy Story" to any of their traditional 2D films, it is obvious that the two methods produce very different effects.

Despite the differences, Disney has used 3D technology to overcome difficult animation sequences. In their recent adaptation of Hercules, the hero becomes engaged in a fight with the multi-headed Hydra. The more he decapitates his rival, the more heads the monster obtains. The sequences becomes very complicated with heads raging in all directions. It became too daunting a task for the traditional animator. Instead a 3D model of the monster was created. The model could now be manipulated to produce the desired animation. However, to enable this 3D animation to blend naturally into this 2D film, the rendering algorithm was toned down to produce the characteristic flat appearance of traditional animation.

However, the differences between the two animation techniques is more than a question of rendering algorithms. Traditional 2D animation is a much freer art form, the limitation of which is the artist's creativity. On the other hand 3D animation is restricted to 3D models complying to real world rules. The subtlety of traditional animation is often in it non-compliance to these rules. A classic example is Mickey Mouse. Mickey's ears are either seen in full profile or face on, there is no inbetween position. Such subtleties cannot be easily modelled.

Therefore there is a need to provide a computer-assisted animation system that specifically caters for the traditional 2D animator. The key issues with such a sys-

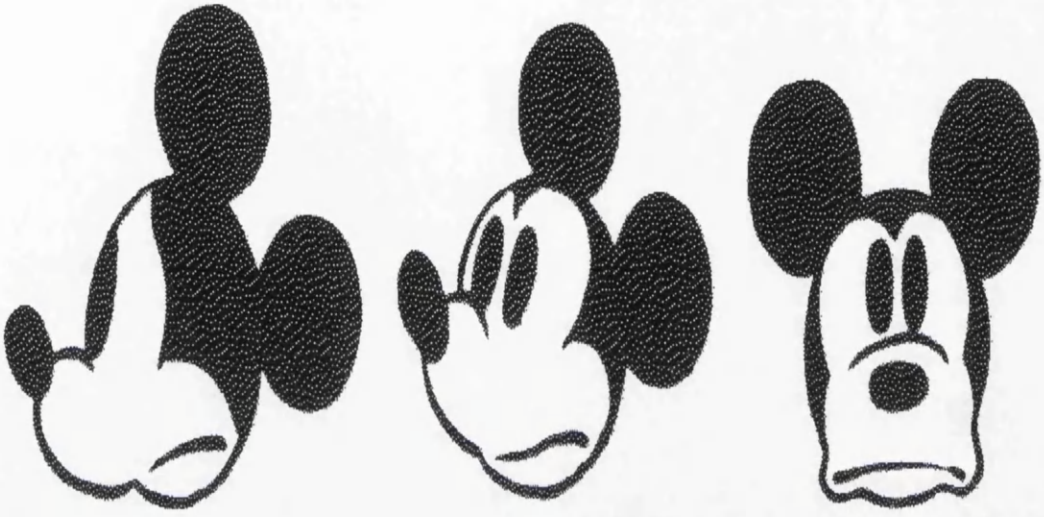

Figure 1.1: Mickey Mouse

tem were identified by Edwin Catmul 20 years ago and still remain issues today [8]. It is not sufficient to develop algorithms to automate the animation process, the studio must also be convinced of the benefit of this new technology. Traditional studios have well established guidelines which must be followed. To computerise this process, the system must respect these guidelines and take into account the overheads the studio will incur in both time and money.

1.2 Overview of Problem Area.

Computer-assisted animation systems are commercially available. The problem is the amount of additional overheads they produce. These overheads are largely due to the ineffective algorithms used to solve certain problems. The thesis aims to minimise some of these overheads by using more efficient methods.

Animators work best with a pencil and paper and is the preferred medium for

creating drawings. At some point, these hand drawn images must be entered into the computer. Here the first problems arise. Scanning is the usual means of transferring the images from paper to computer. However, no matter how good the scanning equipment it is inevitable that some noise with be introduced. The current algorithms used in computer-assisted animation system can not cope with any noise interference. In fact they require the images to be cleaner than the standards expected from the traditional process.

This is particularly relevant in systems which claim to automate the inbetweening phase. Computers must abide by the same guidelines set down by the studio, in order to minimise the overheads. Inbetweening occurs after the sketchy keyframes have been produced but before any cleaning of images has taken place. Therefore the inbetweening algorithms must be able to cope with sketchy images. Insisting on the key frames being over cleaned, places a large overhead on the studios resources. Some means of extracting a cleaned image from the scanned in sketchy image is needed to minimise the overheads produced when converting from paper to computer.

Scanning in images creates another problem. The inputted image is now in raster format. Raster images are so big in size, that in order for a studio to be able to store and manipulate such large amounts of data, expensive equipment is necessary. Not only does this prove expensive for the studio but raster images do not produce as good quality images as their vector equivalents. Vectorising the raster image would greatly reduce the image's size allowing the images to be saved easily and therefore animation sequences can be reused. Current system fail to reliably vectorise the image. The unpredictability of the results means that further overheads are produced to check and correct each image after vectorisation.

Combining these two problems areas, the question is, given a sketchy raster image is it possible to extract a cleaned-up vectorised equivalent with minimum overheads? This is the problem area which this thesis tackles.

1.3 Introduction to Proposed Solution.

Current solutions treat the cleaning-up and vectorisation problems separately. Thinning algorithms are used for extracting core-lines from thicker lines and several design packages already provide the facility to convert from raster to vector. However, these separate solutions do not combine well together with regards to solving the overheads imposed by computer-assisted animation systems.

The failure of these algorithms occurs because they have no understanding of the problem area. They have been developed as low-level process, relying on the image alone for all the information. These global operators are error prone. Much time is spent correcting these errors. Such overheads cancel out many of the benefits in automating the process.

The thesis suggest a more "intelligent" method of resolving the problem of vectorising a sketchy scanned-in image. Active contour models or snakes resolve low-level processing problems but unlike the traditional approach they allow additional information such as a knowledge of the problem area to be used to resolve conflicts in solutions. This provides a more robust system minimising the errors.

The key to snakes is having available prior knowledge of the problem area. The problem area in this case is a sketchy raster image. Problems which occur are how to extract from a group of sketchy lines a single line correctly positioned as the animator intended. Also it must be able to complete vague unfinished lines. If a method for resolving these problems can be formulated than it can be incorporated into the snake algorithm and used to extract the correct complete lines from a group of vague sketchy lines.

Fortunately this is possible. By smoothing the image, sketchy lines merge into one another and the true line runs through the darkest region of this blurred line. Blurring an image and locating the darkest region of a line are easily formulated and therefore can be programmed into the snake's equation. Snakes also have the

ability to extend the endpoints of a line, continuing on the general flow of the incomplete line. This technique is known as snake growing. In addition snakes have a hidden bonus, once the image has been cleaned up, having extracted the true lines and completed vague ones, the result is already in vector form. This is because the snake is itself an energy-minimising spline. Therefore there is no need for a separate vectorisation phase.

1.4 Thesis Organisation.

Chapter 2 studies the guidelines which must be adhered to by an animation studio, and thus the approach which must be incorporated into any computer-assisted animation system. The commercial computer systems which are presently available will then be examined with regard to how they benefit the industry and the overheads they occur. The greatest difficulty in automating the traditional studio is resolving the inbetweening problem. The state of current research in this area is then studied.

Chapter 3 looks at the problem area this thesis will focus on. It addresses some of the overheads produced by the current computer-assisted animation systems, focusing on vetorising scanned in sketchy images. A study of traditional methods for solving these problems is carried out. These methods are then examined in light of the problem. A better solution based on active contour models or snakes is then proposed.

Chapter 4 deals with the wider issue of snakes. It starts by identifying problems with the standard edge detector which brought about the snake technology. A more in depth introduction to the theory behind snakes is then conducted. The problem area is then revisited illustrating how snakes can be used to tackle the problems traditional methods could not overcome.

Chapter 5 suggests possible problems and overheads created by the snake solution. It then proposes further work which may overcome or at least minimise some

of these limitations.

Chapter 6 gives a tabular summary of much of the techniques researched in this thesis.

Chapter 2

Background and Related Works:

2.1 Techniques used in a Traditional Animation Studio.

Traditional animation is a very costly and time-consuming procedure when produced on a large scale. It involves large teams of people working efficiently together. To ensure that the time and cost spent on the film production is kept to a minimum, a tight schedule has been developed to which the large animation studios strictly adhere. Here is an example of a typical production line in an animation studio [20].

- **Script** Whether the story is an adaptation from a book or an original story the first step is to provide a script. In live-action films the dialogue is the most important feature. However in an animated film, the atmosphere is created more by the visual effects than the speech. Therefore the animated script is more concerned with establishing the mood and action of the film and concentrates less on the exact words.

- **Storyboard** Once the basic script has been determined, the director converts it into a series of sketches . Each sketch represents approximately four to five seconds of the film. The whole team of writers, directors, producers and animators are present at this stage. Visualising the story in this way is important so that any deficiencies in the script can be identified and the necessary alterations made. Often where sections of the story prove weak "think tanks" are

formed to make improvements. Working like this enables the team present to appreciate the content of the film.

- **Soundtrack** The visual concept of the film having been set, it is now time to produce the soundtrack. The soundtrack consists of all the music and dialogue for the entire film. In order to obtain perfect synchronisation between the action and the speech, the complete soundtrack must be finished before a single frame is drawn.

- **Track Breakdown** On receiving the completed soundtrack, an editor divides it into workable lengths. For each section an exposure sheet is produced. The exposure sheet consists of a table in which every frame is represented by a single row. The dialogue is broken down phonetically and precisely inserted into the exposure sheet. Each row is completed by adding the animation and camera action associated with that frame. In this way, a row contains a complete set of instructions for the creation of a single frame in the film.

- **Designs** Now that the action for each frame has been determined, the drawing can start. But first each of the main characters must be created. A single designer is usually assigned to an individual character. When the interpretation of that character has been approved, the designer produces a model sheet. The model sheet shows the character in a wide range of poses, viewed from different angles and displaying a variety of expressions. It indicates clearly the interpretation of that character, setting its mood and displaying its intended movements and behaviour. As many artists will be assigned the task of drawing this character, the model sheet acts as a very important reference manual for maintaining consistency of character throughout the film.

- **Leica Reel** Now that the characters have been created the original scribbles of the storyboard are redrawn by the layout artist. The correct size and style

Figure 2.1: Model Sheet

of each character, as obtained from the model sheets, is now incorporated into the pictures. These accurately drawn frames are then shot onto film. As no animation has yet been produced, the Leica Reel consists of a sequences of stills. Each still is held for precisely the length of time its corresponding animation will take, according to the exposure sheet. The Leica Reel is then played alongside the soundtrack. From this a good impression of the overall film can be established, flaws can be corrected and all redesigning finalised. After this the animation will start and any changes to the overall visual content of the film will incur large losses in both time and cost.

- **Drawing** The Leica Test having been approved, the actual line drawings of the animation can begin. For each sequence of film a hierarchical team of animators is set to work. The main animator produces a sketchy sequence of extremes. The purpose of these extremes is to depict the action. The assistant animator takes these extremes and breaks them down into key frames.

Typically key frames consist of every fifth frame of the animation. It is the inbetweeners' task to draw the remaining frames missing form between these key frames. The inbetweener is the most junior animator working on the line drawings [16]. Each frame of the film has now been drawn using a pencil and paper.

- **Line Test** As each sequence of line drawings is finished, it replaces the corresponding stills in the Leica Test. Filming is carried out precisely according to the exposure sheet. This ensures that the movement and interaction of characters is as intended. Any errors detected must be reworked and the sequence reshot. It is extremely important that any problems are ironed out at this stage as alterations made later may prove very damaging.

- **Cleanup** Taking the completed line drawings, the team of cleanup artist must ensure that it is visually consistent throughout the entire film. With so many individuals working on a single character, it is necessary to check that no differences in personal styles have slipped in during the production. Having redrawn characters to correct the variation in appearance, another line test is performed to find any new errors that may have been introduced.

- **Trace and Paint** The animation is now ready to receive its colour. But first each line drawing must be transferred to a thin sheet of celluloid or acetate known as a cel. It is usual that a single frame consists of more than one cel. Overlapping of cels requires that opaque ink is used, thus obscuring any drawings from lower cels. Although in certain cases translucent ink is used to achieve certain effects. Once the painting is completed, the animation is ready.

- **Backgrounds** While the animations are being painted, another team of artists is busy preparing the backgrounds. Their style has been determined during the design phase. The background consists of anything in the frame, either

behind or in front of the characters, which isn't animated. Because it remains constant it often appears for long stretches in the film. Therefore the artists can afford to spend more time on a single background which is often more detailed than the characters themselves. All the components needed to create a frame have now been finished.

- **Checking** As each frame is completed, it is passed to the checker. The checker must do a thorough job in identifying any errors in the actual painting or in the interaction between characters and backgrounds. This is an extremely important stage, after which filming will commence.

- **Final Shoot** When the checker is satisfied, the artwork is passed to the cameraman to be shot onto film. The cameraman brings together the artwork for each frame on the rostrum camera, illuminates it and shoots. Rear and forward lighting is available and for the illusion of depth a multiplane camera may be employed. This places sections of the art on different planes depending on their relative depth in the scene. It enables the camera to zoom in on a scene, enlarging the foreground objects as desired while maintaining the size of distant objects. A complete colour version is available for the first time on film.

- **Rushes** The first viewings of this colour film are known as rushes. In its completed form it is now finally examined for any remaining errors. Any problem must be corrected and reshot. As each section is passed, the editor takes the colour version and replaces the corresponding line drawings in the Leica Reel. This completes the filming stage.

- **Dubbing** Once the visual content of the film is ready on a single roll of film, the director and editor must finalise the audio content. The music and dialogue have been available even before any drawing could take place. What remains to be done is to choose the sound effects and lay them in perfect syn-

chronisation with the corresponding action in the film. In the dubbing theatre the sound effects along with the music and dialogue are combined onto one soundtrack. The film now exists in a double-head stage, with the soundtrack and visual content on two separate rolls of film.

- **Answer Print** From the double-head stage the editor orders an answer print. This involves taking the picture and sound and combining them onto one roll of film. One final intensive checking of the entire film is performed. The animated film is now ready.

The production of an animated feature film is by no means a trivial matter. For a large animation studio, not adhering to these strict guidelines can be extremely expensive in terms of time and money. Any attempt to computerise the traditional process must also respect such rules. It is not sufficient for the computer to be able to mimic a set of tasks described here, it must be able to do so with minimum overhead. Computers will be of little benefit to a studio if this well established process must be altered in any way to accommodate the computer or if large teams of extra staff are required to operate it. All their possible advantages would be outweighed by the addition costs occurring. Computers will not be adopted into the traditional animation studio unless they are designed to fit in easily. Examining these guidelines is therefore of paramount importance for the success of any computer-assisted animation system.

2.2 Areas of 2D Animation Already Computerised.

Three types of computer-assisted animation systems are commercially available at present [10].

1. Ink and Paint Systems.

2. Automated In-Betweening Systems.

3. Paperless Systems.

Each of these systems is based on the traditional animation studio guidelines. The difference between them is the number of guideline stages which they claim to computerise. A comparison will now be carried out between each system stating,

- the degree to which it has been computerised,
- the benefit it brings to the animation process,
- the overhead it occurs.

2.2.1 Ink and Paint Systems

The Ink and Paint system is the most modest in its claims. As its name suggests, it is responsible for inking in the line drawings and then painting each cel. All other processes prior to the painting stage are carried out in the traditional manner. The system takes over when all the line drawings are available. As well as inking and painting, this system can also record the final animation on film or video. This is the oldest computer-assisted animation system. Two of the most well known products are PEGS and Toonz.

1. Claims

   - Noise is removed and small gaps are closed and the image is generally cleaned in preparation for the painting step.
   - The image is painted using a seed-fill algorithm.
   - The animation and background is composed to produce each frame.
   - Shooting is carried out using a virtual rostrum camera which can zoom, rotate and pan as well as providing a multi-layer system.
   - Recording can be done on film or video.

2. Advantages

   Hand painting every cel of a feature film is extremely tedious task and probably the most time consuming. Not only must this be a done very accurately but it is also important to mix the right colour each time. Ink and Paint

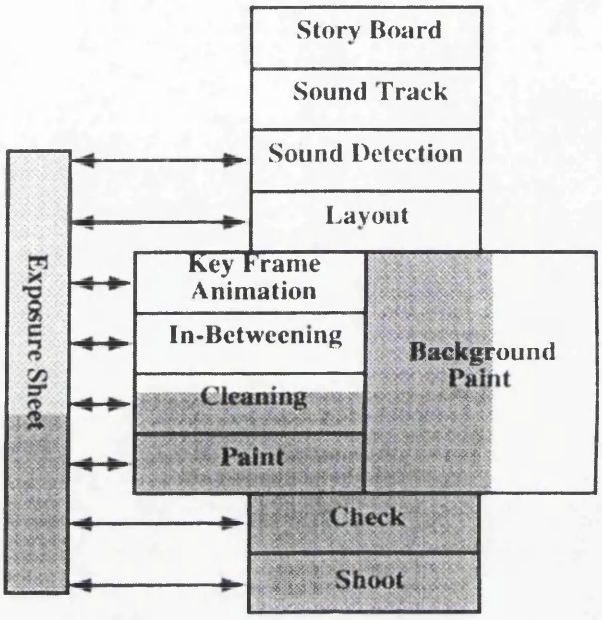

Figure 2.2: Ink and Draw

Systems have been extremely successful in automatically this work. Not only does the seed-fill ensure that only areas within a region get coloured, but the computer offers the artist an extensive range of colours which guarantee to be the same shade every time.

More sophisticated painting algorithm have matching techniques built into them [11]. Matching between successive frames enables the colour to propagate from one frame to the next with no human intervention except where the algorithm fails to find a match. Such methods further speeds up the painting process.

Another advantage of using the computer over traditional painting methods is that the computer can cope with larger numbers of composition layers. The traditional method was limited to five layers, after this the acetate celluloid becomes too opaque.

Filming the animation is also made easier with the computer. Traditionally a rostrum camera is used which consists of a camera and several movable transparent trays at different levels to give the illusion of depth and finally the background layer. The camera could pan the scene and zoom in and out and the cels can also be moved and rotated as required. The computer's virtual rostrum provides the same facilities but is much easier to set up and use.

Backgrounds can be generated from various sources. Usually the artist paints a high quality image and scans it in. However it is also possible to use computer generated images or live action film.

3. Disadvantages

Even the best scanning equipment introduces noise into the image. If a gap appears in a line as a result of noise, the seed-fill algorithm will bleed one area's colour into the next. This requires extra staff to check each image. To avoid having to spend large amounts of time cleaning the raster images, the animator is required to produce cleaner drawings. All this over-cleaning, scanning and checking adds greatly to the studio's overheads.

Raster Images occupy too much memory space to enable them to be stored for further use. Except for the background scenes, images are usually removed from the computer system as soon as the frame is finished. The technology required to store and manipulate such a large amount of data adds great expense to an animation studio's budget.

Not only do raster images occupy a lot of memory but zooming causes them to become pixelated. Such deterioration in image quality cannot be tolerated especially as zooming is often used when filming. To overcome this problem the raster images must be scanned in at high resolution, thus creating even

larger image sizes.

2.2.2 Automated Inbetweening Systems

The Automated Inbetweening System is an extension of the Ink and Paint system. It attempts to commence the computerisation process after the keyframes have been produced. This implies that it is now the computer's responsibility to produce the inbetween frames. Once the inbetween frames have automatically been created the system progresses in a similarly manner to that of an Ink and Paint system. Animo is an example of an Automatic Inbetweening system.

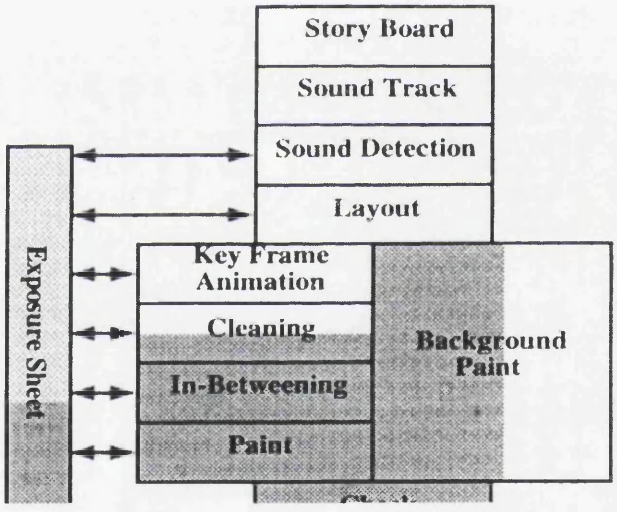

Figure 2.3: Inbetweening Systems

1. Claims

   - It cleans and vectorises the scanned in key frames using semi-automatic methods.

- Objects in the key frames are matched with adjacent key frames.
- These matched objects are then interpolated according to the exposure sheet to produce the inbetween frames.
- Painting can be done using a seed-fill algorithm or by matching it to a template or previous frames.
- The animation and background is composed to produce each frame.
- Shooting is carried out using a virtual rostrum camera which can zoom, rotate and pan as well as providing a multi-layer system.
- Recording can be done on film or video.
- Sequences of animation can be saved in a database for reuse.

2. Advantages

As the inbetween frame will be automatically generated by the computer, only the key frames need to be hand drawn. This greatly reduced the number of images which must be scanned in and cleaned.

One of the benefits of vectorising an image is that it is scale independent, allowing the image to be zoomed to any degree without disintegrating, and avoiding the need to scan at high resolutions.

The vectorised image also requires much less space when stored which enables a database to be set up to reuse animation sequences, further reducing the amount of artwork which needs to be hand drawn and scanned in.

Being able to match successive key frames makes painting easier. As correspondence between regions has been established in the inbetweening process, the same correspondence enables propagation of colour to successive frames. In addition, if an area is patterned, for example a tartan kilt, than as the character moves the inbetweening will assure that the tartan will move correctly.

Along with these advantages the automatic inbetweening system also provides all the benefits associated with ink and paint systems.

3. Disadvantages

The vectorising algorithm currently employed by Animo is too unreliable. The amount of checking and correction required after vectorising creates too great an overhead to make this a feasible process.

Correspondence matching still remains a very difficult problem to resolve for all but the simplest cases. Often the only means of correctly establishing a match is by human intervention. Such an approach is extremely time consuming and would require extra staff to carry out the task. The overheads would be enormous.

Inbetweening still remains a very difficult problem to resolve. The algorithms that have been developed are very limited in the movement they can generate. Too limiting to create the majority of inbetweens. So until there is some breakthrough in inbetweening, most line drawings will have to be done by hand.

2.2.3 Paperless Systems

Most of the overheads created by the previous two systems were due to the change of medium. Scanning in the hand drawn images introduced many of errors into the process. Therefore many of time and staff is needed to check and correct the computer version. To overcome this problem, the paperless system does not have any change in medium. All the work, except the creation of the sound track, is produced directly on the computer. The first commercial paperless product is TicTac Toon.

1. Claims

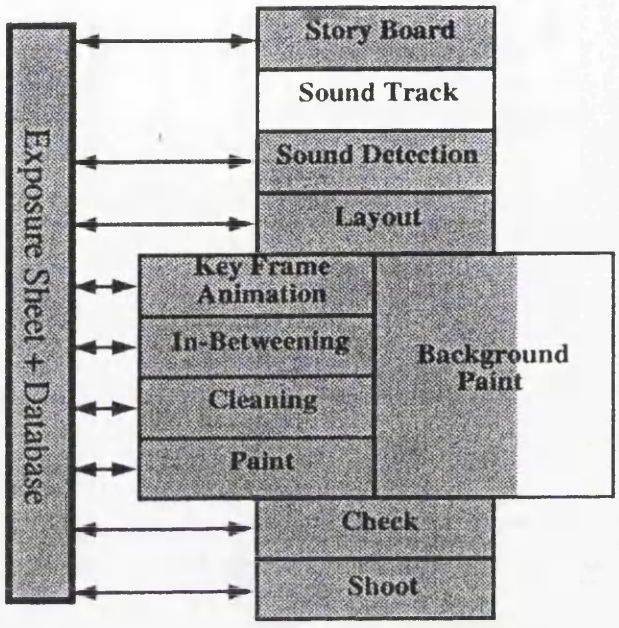

Figure 2.4: Paperless

- It provides a facility for designing the storyboard and Leica reels directly on the computer.
- Each frame is drawn directly into the computer with the aid of computerised tools similar to those used in the traditional studio, such as a lightbox.
- Each line is converted into vector format as soon as it is drawn. By varying the pressure on the pen, the width of the line varies. The line's width is preserved in the vectorisation process.
- Close gaps in lines are automatically closed.
- The image is painted using a seed-fill algorithm.
- As new artwork is produced the rough sketches of the Leica Reel can be replaced by the new work.
- The animation and background is composed to produce each frame.
- Shooting is carried out using a virtual rostrum camera which can zoom, rotate and pan as well as providing a multi-layer system.

- Recording can be done on film or video.
- Sequences of animation can be saved in a database for reuse.

2. Advantages

   Using the computer from the start avoids the overheads produced when scanning in and cleaning up images.

   All the images are drawn directly onto the computer and automatically vectorised, making the images resolution independent.

   Vectorised image require much less storage space.

   Varying line thickness can be captured by altering the pressure asserted on the pen.

   Having each stage of the animation process available on the computer has great advantages when checking the animation for inconsistencies. Updating the leica reel with new artwork becomes a trivial matter of substituting the old for the new. Not only can the animation be checked for errors in the drawings, but the soundtrack is to hand to give a clearer impression of the film's overall progress.

   Along with these advantages the automatic inbetweening system also incorporates all the benefits associated with Ink and Paint systems.

3. Disadvantages

   Although this system claims to have computerised the whole traditional process, up until the painting stage, it is really just substituting the paper and pencil for the tablet and screen. All stages before that are carried out in exactly the same manner but with a different medium. The only advantage

is that as the animation is created it can easily and speedily be checked for errors. Still each line drawing must be created by hand, with the disadvantage of using unfamiliar tools to achieve the same effect.

Most developers of computer-assisted animation systems disregard anything that would limit the artist's creative freedom. It is also acknowledged that at present there does not exist a input device that satisfies the artist's needs. Therefore the earliest stage at which computers should be introduced is after the key frames have been hand drawn. This allows the artist use the conventional tools of a pencil and paper. Paperless systems, in their current state, are therefore not suitable for the large traditional studio.

Most system developers reckon that the paperless system is not a feasible system. Introducing unfamiliar tools with which to create the drawings is considered too great an overhead for any studio.

The Automatic Inbetween system has the right approach, scan in the key frames, vectorise them and then interpolate between them to produce the inbetween frames, after which it can carry on like the ink and paint system. Unfortunately the vectorisation algorithm employed by Animo produces unpredictable results which have to be carefully checked and corrected. Inbetweening algorithms still remain very basic and wrought with problems preventing the system from creating all but the simplest of movements. To illustrate the difficulties involved in automating the inbetween process, the next section will examine the current state of inbetween research.

As the vectorisation and inbetweening processes are unreliable at present, the Automated Inbetweening system reduces to little more than an Ink and Paint system. Though the latter system does incur overheads, the reliability of its algorithms make it the most popular and well known computer-assisted animated system.

2.3 Inbetweening Techniques.

At first inbetweening appears a simple problem. The assistant animator produces key frames which are every four frames apart. It is the inbetweeners' task to complete the action by filling in the missing three frames between a pair of key frames. As each frame represents $\frac{1}{24}th$ of a second of the film, the amount of change between one frame and the next is normally very small. Yet the inbetweener must redraw the whole character each time. Such time consuming work would seem to be a job the computer could easily replicate. Automatically producing three quarters of all the line drawings for a film is a very tempting prospect. As a result much research has been carried out in the hope of achieving this goal.

However Catmul points out [8] "This problem might be likened to another well known problem, given Russian and English, find the correspondence between them such that we can transform a sentence from one language to the other. On the face of it the problem seems quite simple but we know that it isn't. Similarly there are subtle problems with inbetweening which require intelligence to resolve."

Human inbetweeners have one big advantage over their computer equivalents, they are usually familiar with the object they are animating. Familiarity with an object provides the inbetweener with lots of vital information such as the three dimensional structure of the object and how various parts of the object move in relation to each other. Such knowledge enables the inbetweener to produce a convincing sequence of movement.

What the computer sees is a two dimensional representation of this object. There are no clues to its overall structure. This is insufficient for most animation sequences. Even a common gesture such as a turning head causes major problems for the computer. What happens if an ear suddenly appears and how does the moving profile of the nose effect the eye? Such information can not be inferred from the two dimensional keyframes.

Along with this built in knowledge, humans can often reference model sheets of the characters they are inbetweening, providing yet more clues to how the character moves. In addition the assistant animator sometimes indicates the desired motion on the key frames themselves. However it is interesting to note that despite all this information, humans also make mistakes.

Despite their clear disadvantages, computer algorithms have been developed in an attempt to automate the inbetweening process. This research appears under several names such as inbetweening, interpolating, shape blending, image transformation and morphing. Each has its own degree of success and shall now be discussed.

The simplest means of inbetweening two key frames is based on linear interpolation. This technique is known as simple linear interpolation or cartesian coordinate linear interpolation. Linear interpolation works on the basis that key points, such as corners, in the first keyframe are identified. These are then linked to their new positions in the next keyframe. These pairs of points in the two frames are known as corresponding points. If three frames are required between successive key frames, an imaginary line is drawn between each pair of corresponding points. The line is then divided into quarters, where the first quarter mark represents the point's new position in the first inbetween frame, the halfway mark indicates position in the second inbetween frame, and so on.

When this process is completed for all corresponding points, the newly generated points for a single inbetween frames are linked together once more. This is a very simple technique and works very well if the movement is totally linear. Unfortunately for most movements this is too restrictive as the majority of actions are not limited to linear motion or timing.

To capture this non-linearity in motion and timing, the human inbetweeners often require extra information. When this occurs the assistant animator draws a trajectory from a point to indicate the path it will take to reach its corresponding point in the next keyframe. The timing may also be determined by marking the

inbetween frames' positions along the trajectory. Closely spaced marks result in a slow progression from one frame to another. This technique gives the animator more control over the animation and the result is less rigid.

Reeves [24] incorporated this method into the linear interpolation approach in order to overcome its restriction to linear motion and time. Paths could now be constructed between corresponding points to determine the path and speed of the motion. A patch network of paths, with the optional timing spacings, is drawn from one keyframe to the next and linear interpolation is used to generate the required positions of the inbetween frames. This set of keyframe constraints is known as Moving Points Constraints.

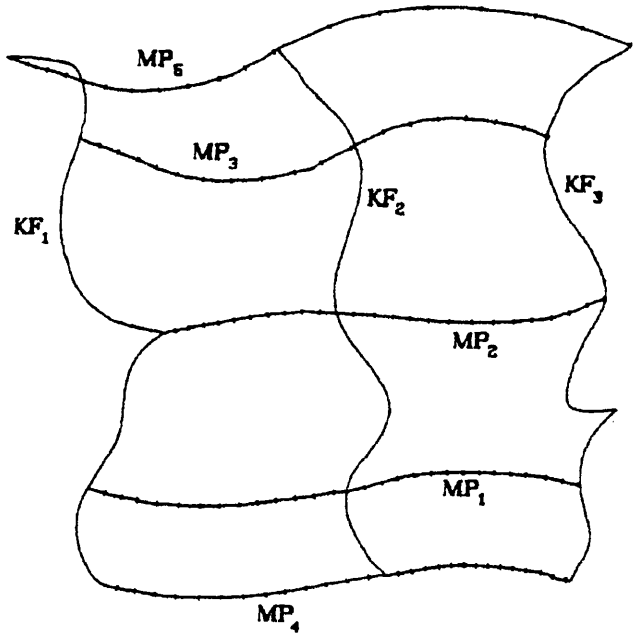

Figure 2.5: Moving-Points

So far both techniques have relied on interpolating between a pair of cartesian co-ordinates. Independently interpolating each pair, with no knowledge of the neighbouring pair, results in the lengths of lines and the angles between neighbouring points not being preserved. Therefore the inbetween frames often don't make

any sense, especially if any degree of rotation is involved.

Alternatively, instead of representing an object by its cartesian co-ordinates, it can be described by its polar co-ordinates [25]. One point is chosen to be the anchor point and all other points are described relative to its neighbour in terms of angles and lengths. Using this technique, if the transformation from one keyframe to the next requires rotation the relative angles and lengths of the sides of the object can be preserved.

Sederberg developed a method based on polar co-ordinates and called it 2-D Shape Blending [21]. Every object must first be described in terms of its polar co-ordinates before any correspondence between key frames is established. Now the interpolation is performed on these angles and length and not the cartesian co-ordinates. Unfortunately this technique often results in an unclosed object. This is overcome by forcing the first and last points to coincide and then tweaking the intermediate angles and lengths slightly and in proportion to their relative sizes. [22].

The techniques discussed so far rely entirely on the silhouette of the object to obtain correspondence points. Each is error prone. Even Sederberg's shape blending, despite his optimisation algorithm to ensure that the generated objects remain closed, still fails to prevent self-intersections. This is where lines defining the outline of the object cross over other parts of the outline.

Goldstein proposes that the reason why the interpolation of the silhouette fails is that it does not take into account the geometry of the object or its internal area [9]. He identifies three rules that must be observed if the inbetween frames are to be correctly generated.

1. If both key frames are closed and simple, the inbetween frames should also be closed and simple.

2. The area of the generated shapes should blend smoothly from the area of the first keyframe to the area of the second.

3. If the second keyframe is a translation or rotation of the first then the inbetween frames should also be the appropriate translation or rotation.

Goldstein achieved these three rules by first calculating a multi- resolution representation of the object to be interpolated. This technique is known as curve evolution. Multi-resolution is obtained by smoothing an image using a Gaussian filter. The resulting resolution depends on the width of the filter, the wider the filter the lower the resolution. A smooth closed curve will eventually converge to a point if it is repeatedly smoothed using a Gaussian filter of increasing width.

Points along the surface of the object are traced and the paths taken between each resolution are mapped. This is called the geometric evolution path. The number of iterations a curve takes to converge to a point is called the depth. In order to be able to interpolate between two key frames the depth of both must be equivalent. This is easily achieved.

Interpolation between successive key frames starts at the lowest resolution, that is the centre point. After each successive interpolation the process moves out one depth along the geometric evolution path to the next highest resolution and performs a further interpolation between the key frames. The reasoning for this is that the low resolution captures the global transformation needed, such as rotation or translation, to capture the overall movement of the object. Whereas the high resolution depicts the fine local detail of the object which may need slight adjusting in the final stages.

Ranjan also saw the need to capture the internal geometry of the objects [23]. This was achieved using technique known as the union of circles. With this method the internal shape of the objects is filled with overlapping circles, so that these circles fully describe the objects.

To interpolate between union of circle representations, the two objects are first aligned. If necessary they can be automatically segmented and the correspondence between the various parts of the two objects established. Matching up the circles

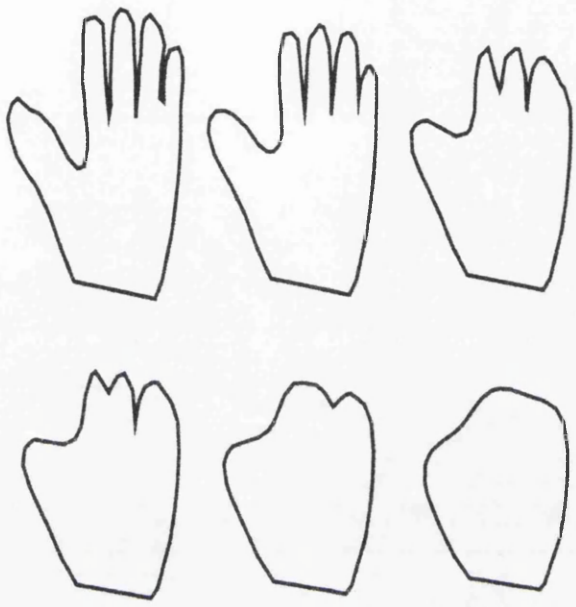

Figure 2.6: Multi-resolution

within these parts calculates the required transformation from one object to the other. After each step of the procedure, the separate parts are integrated back into a whole object.

Although the last two techniques take into consideration the internal geometry of the objects, they are still considered shape based forms of interpolating. Burtnyk took a different approach based on skeleton interpolation. Traditional animators often replaced a complex object with a stick figure representation in order to work out the movement. Once the animator is satisfied with the motion of the stick creature the more complex features of the character are replaced. Burtnyk took a similar approach. He stressed that this technique was not suitable for most inbetweening but only where the motion was complicated.

With this method, not only have the key frames to be hand drawn but also the stick figures of the objects in these frames. The objects are then replaced by their skeleton representations and interpolation between these is performed. If the user

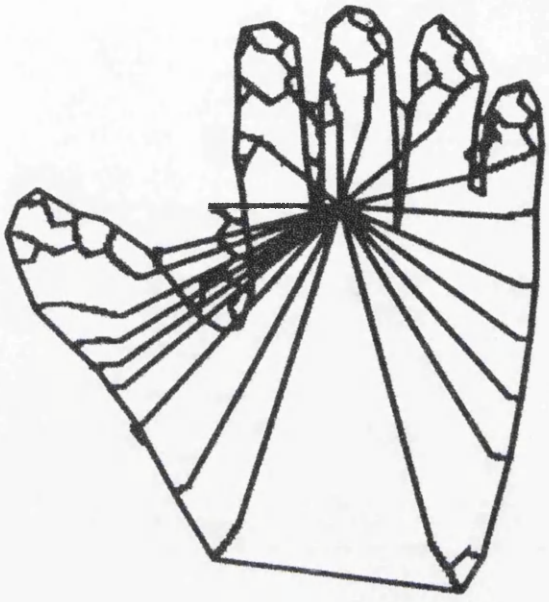

Figure 2.7: Evolution Path

wishes the motion to be more complex than the automatically generated inbetweens, the skeletons can be manipulated by the user. This causes the inbetweens to adjust accordingly. Having achieved the desired effect the outer skin of the stick figure is replaced to complete the inbetween frames.

Shapira extended the idea of skeleton-based interpolation [14]. A star-skeleton representation is used to describe the geometry of both the interior structure and the boundary of the object. Unlike Ranjan's work, this method is to be used for all inbetweens and the skeletons are not hand drawn but developed with the aid of the computer.

To obtain a star-skeleton representation of a complex polygon, it is first divided up into individual simple polygons. The midpoint in the dividing line between each of these simpler polygons is connected to a star point within that polygon. A star point is a chosen point which can see every other point in that polygon. A polygon nearest the middle of the whole shape has as its star point a vertex of the polygon.

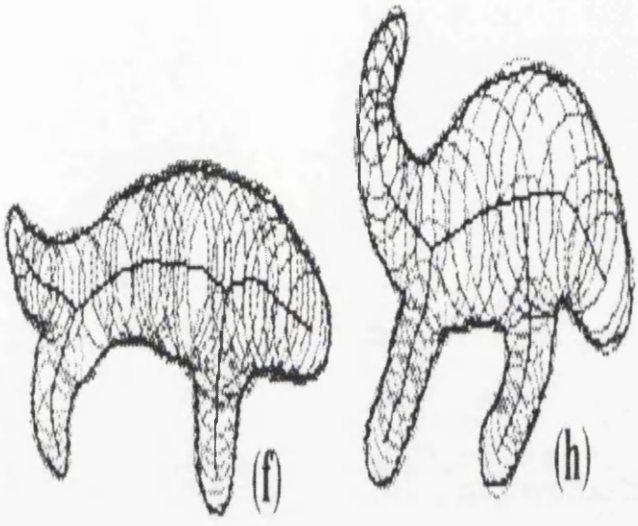

Figure 2.8: Skeleton

This is known as the root. A path must lead from the root to every star point and midpoint in the shape. This network of paths forms its skeleton.

Having established the skeleton, the polar co-ordinates of every point within each polygon is calculated relative to the star point. These new values are used to interpolate between objects, but often it is necessary to add extra points when matching the skeletons. As every point on the boundary and within each polygon is expressed in terms of the star point and every point linked to a single root, there exist a great awareness of the shape of the object as a whole. This knowledge of structure prevents many of the errors of self-intersection and non-closure associated with other techniques. The inbetween frames generated using the star-skeleton algorithm have very good results.

Despite the good results achieved using several of the described methods, automatic inbetweening algorithms are still in their infancy. With the exception of Burtnyk's interactive skeleton representation, none of these techniques achieve any-

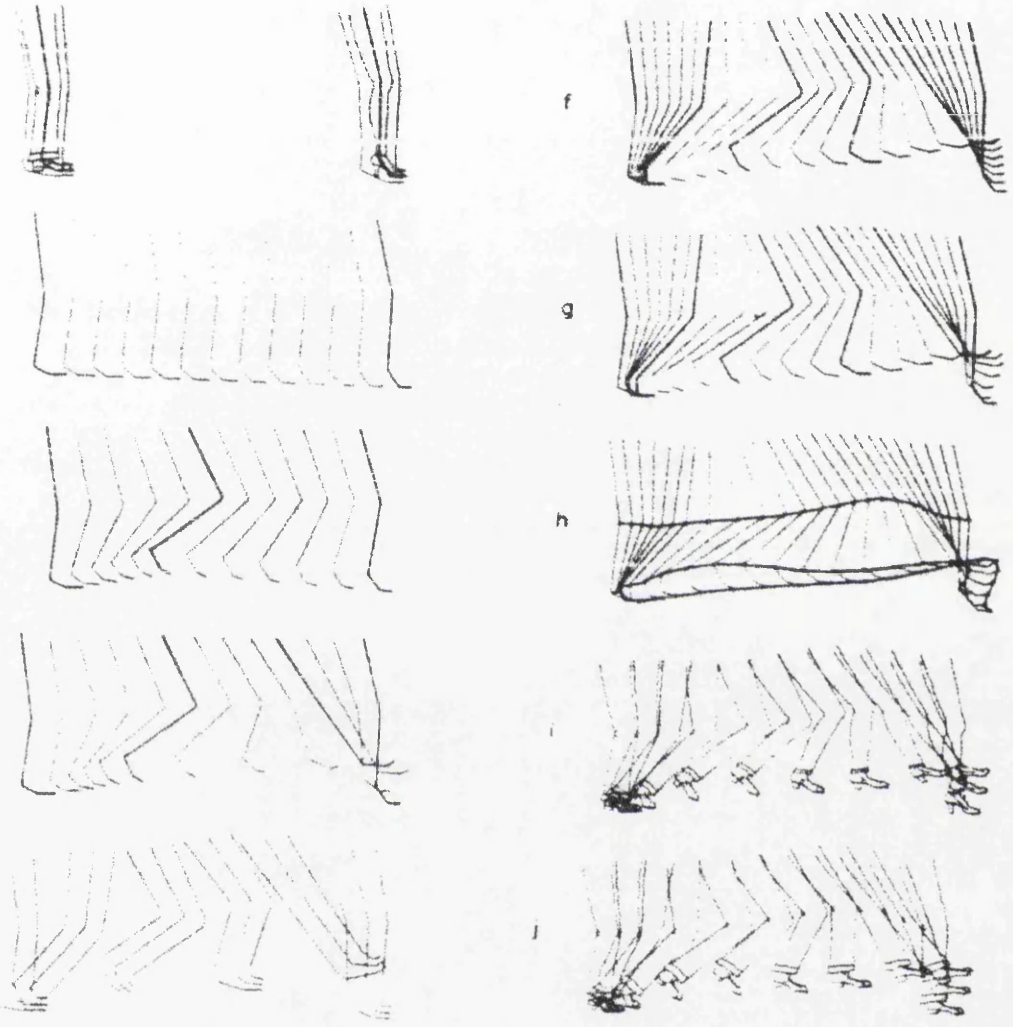

Figure 2.9: Leg Sequence

thing more than a smooth blending from one shape to the next. Burtnyk's approach was developed using traditional animation ideas but requires too much user intervention to be of any benefit.

The idea of completely replacing the human inbetweener with a computer system still remains an illusive one. But this does not mean that the computer can not be used to automatically generate inbetween frames. Simple tasks, such as animating a bouncing ball, are easily produced by the computer. Such aids should be built into a computer-assisted animation system to relieve the artist from having to hand

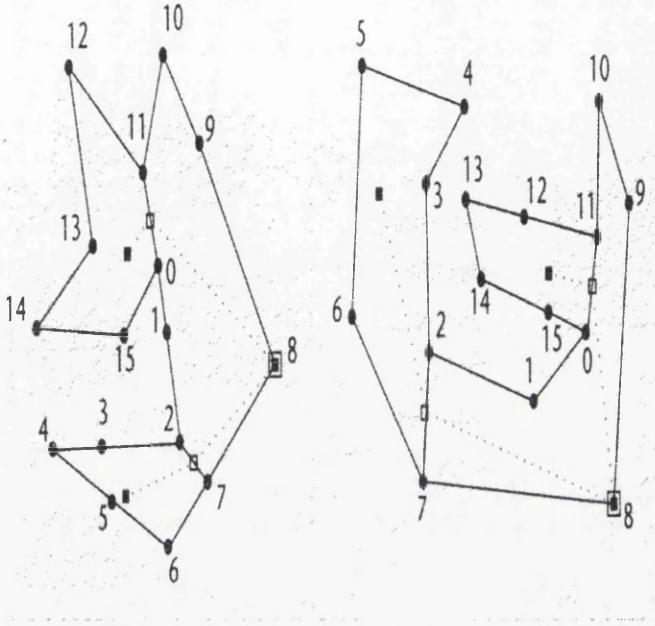

Figure 2.10: Star

draw frames where the transformation between each is straight forward. But until there is a major breakthrough, complex movement will still have to be done manual using traditional methods.

Chapter 3

Problem Area and Proposed Solution:

3.1 Restatement of the Problem Area.

The introduction of computer-assisted animated systems will not occur unless the computer systems can fit into the guidelines set out by the traditional method and produce as good quality results with a minimum of time and financial overheads. Taking the computer-assisted animation systems presently available, the overheads produced by each will be re-examined.

- Ink and Paint System.
    1. Over-cleaning of frames prior to scanning and the removal of noise and errors from the scanned images.
    2. Storage and manipulation of the large raster images produced by scanning.
- Automatic Inbetweening System.
    1. Over-cleaning of frames prior to scanning and the removal of noise and errors from the scanned images.
    2. Human intervention in correcting mistakes caused by vectorisation process.

3. Limitation in the effects produced by current inbetweening algorithms.

- Paperless System.

1. change in medium from pencil and paper to tablet and screen.

The paperless system, though producing just one overhead, will be ignored. The consequences of replacing the animators' traditional tools is considered too drastic an action to be contemplated any further. To employ this system in a studio would require the complete retraining of all the staff. Such changes could seriously restrict the creative nature of the animation produced. Therefore the overheads produced by this system far outweigh any benefits it may have.

Another overhead which shall not be tackled in this thesis are the problems associated with automatic inbetweening. The last section of the previous chapter revealed the limitations of the current algorithms. Such a problem still remains the largest hurdle in producing a complete computer-assisted animation system. However, there are some suggestions in the Further Work section which indicate how the technique developed in this thesis may be used as an aid in the production of automatic inbetweens.

There now exists three overheads to be minimised.

1. The over cleaning of images prior to scanning.

2. The storage and manipulation of raster images.

3. The vectorisation of scanned-in images.

The first problem arises because the computer requires the images it works with to be cleaner than those expected by an animator. The problem is much more serious when dealing with the automatic inbetweening system. To reduce the overheads produce by the change in medium, the computer system must slot into the correct position in the production line with a minimum amount of effort. The inbetweening phase comes directly after the production of the keyframes. At this stage in the proceedings all the artwork is in the form of sketchy line drawings. The cleaning-up

phase does not occur until after all the animation has been completed. Therefore the system must also be able to cope with incomplete sketchy drawings.

The other two overheads can be considered as just one, that of vectorising scanned-in images. If the raster images of the ink and paint system are vectorised after scanning the problem of storage and manipulation is reduced. Vector images require much less storage space than their raster equivalents. Also they have the added bonus of being scale independent. This means that the image does not become pixelated on zooming, thus making high resolution scanning unnecessary.

The remainder of this thesis will address these overheads. The problem can be summarised as follows: given a sketchy raster image, does there exist an algorithm which can produce the cleaned-up vectorised equivalent?

Extracting core-lines from broader lines has already been achieved by thinning algorithms developed in image processing. Various techniques exist, each one takes a thick line or object and produces a skeleton line running through it. This is the method used by the current automatic inbetweening system.

Likewise, there exists in computer graphics a multitude of algorithms to vectorise a raster image. These techniques are more commonly known as curve fitting. As the name infers, it involves fitting a smooth continuous vector curve to a raster line.

The next section will examine the algorithms currently available to solve this problem area.

3.2 Current Solutions.

Both thinning and vectorising algorithms have achieved a fair amount of success in their applied areas. Thinning is usually associated with object descriptors. It is the

first step towards classifying an object. Such algorithms reduce an object's lines to a width of one or two pixels. From these skeleton representations a description of the object is derived. Often the information necessary to reconstruct the original objects is stored along with the skeletons. Such additional knowledge is unnecessary in the case of thinning for computer-assisted animation.

A thinning algorithm requires that a line several pixels wide is replaced by a continuous line with a maximum width of two pixels and in some cases it is necessary that the result is no more than one pixel wide. Most algorithms consist of stripping away the outer boundaries of the line until all that remains is a continuous line of the desired width. The following algorithm reduces a line to a width of just two pixels and is typical of many thinning algorithms [15].

$$R_{new} = S(R_{old}) \cup [H_o(S(R_{old})) \cap R_{old}] \cup [R_{old} - H_i(R_{old})]$$

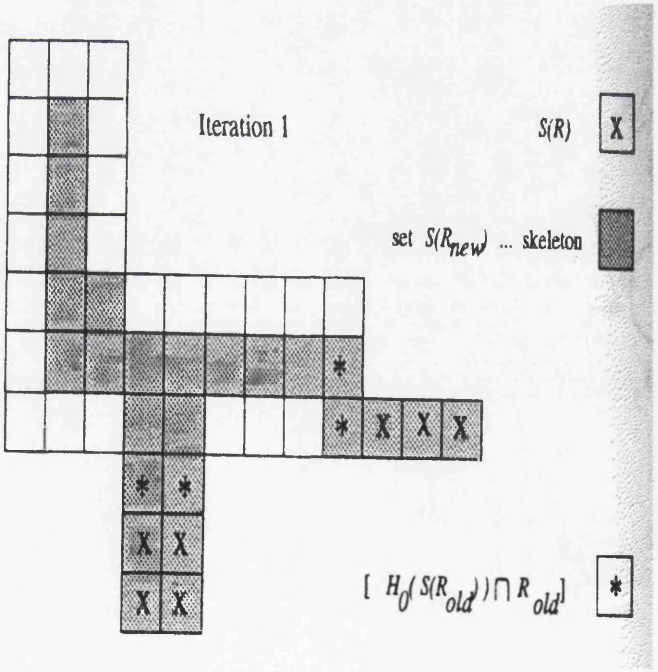

Figure 3.1: Thinning

- where R_{old} represents all the pixels in the original line,

- $S(R_{old})$ is the set of pixels contained in R_{old} with 8-connectivity to pixels either from the background or the boundary of the line, i.e. all the neighbouring pixels of this set form part of the background or the boundary of the line,
- $[H_o(S(R_{old})) \cap R_{old}]$ are all the pixels in R_{old} with 8-connectivity to the pixels in $S(R_{old})$, i.e., the pixels of the line that are neighbours of the previous set.
- $[R_{old} - H_i(R_{old})]$ are all the pixels in the line that do not form part of the boundary.

The algorithm strips away one pixel from the boundary of the line, saving those pixels where the line is at most two pixels wide. This process is repeated with R_{old} being assigned the pixels of R_{new} after each iteration, until $R_{new} = R_{old}$. When this occurs no more stripping is possible and the line consists of pixels a maximum of two pixels wide. R_{new} contains a skeleton representation of the line.

Another method of retrieving the skeleton form of an object is medial axis transforms [15]. A pixel belongs to the skeleton if it is the minimum distance from at least two boundaries. In other words, from every point on the skeleton a circle can be drawn whose boundary touches at least two boundaries of the object. The radius of these circles can be stored enabling the object to be reconstructed from its skeleton form. Unfortunately medial axis transforms are very sensitive to small changes in the shape of the object and noise along its boundary can greatly effect the resulting skeleton.

A variation on the medial axis transform is the maximal square moving algorithm, designed to trace core lines [19]. Medial Axis Transforms work by finding the largest disk to fit into the line at every point. It was developed for the continuous image and the result is a skeleton of neighbouring pixels. Maximal square moving as it name suggests operates by growing a square to its maximum size along the line. It functions on the raster image and the skeleton comprises of the midpoints of each neighbouring square linked together.

Most thinning algorithms require that the whole object is present before thinning

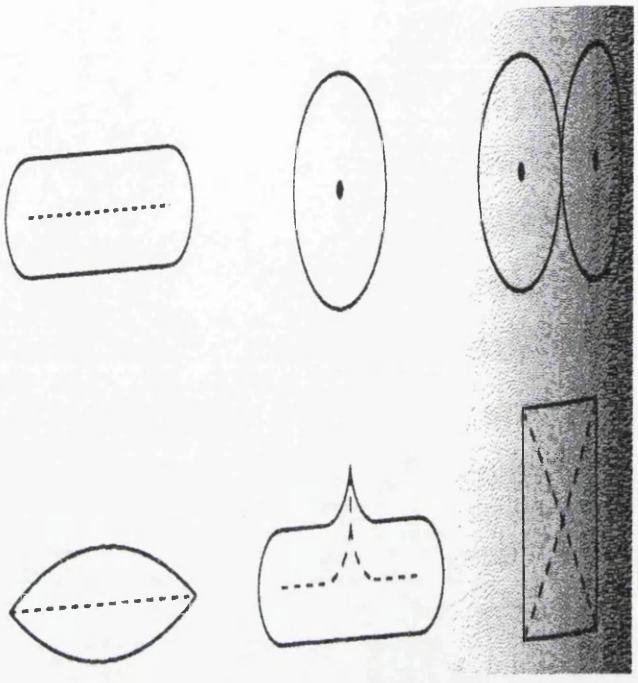

Figure 3.2: Medial Axis

can commence. The maximal square moving algorithm can construct the skeleton as the object is being inputted row by row, thus avoiding the need to store the whole image. Figure 3.3 illustrates how the maximal square moving algorithm operates. A square is represented by an ordered pair $((i,j),l)$ where (i,j) is the coordinate point of the upper left corner of the square and l denotes the length of the square. The centre of a maximal square is indicated with a $\bullet$.

When the first row is a square at point $(6,1)$ is initiated. It is represented by the ordered pair $((6,1),0)$ as the current length of the square is zero. After the second row this square grows to be $((6,1),1)$ and a new square starts at $((1,2),0)$. Both this squares are further enlarged after the third row to $((6,1),2)$ and $((1,2),1)$ respectively. Neither square can increase any further when the fourth row is input so they are both at a maximum size.

When a square reaches its maximum size, pixels to the left, right and bottom of this square must be checked to locate any possible new squares bordering this max-

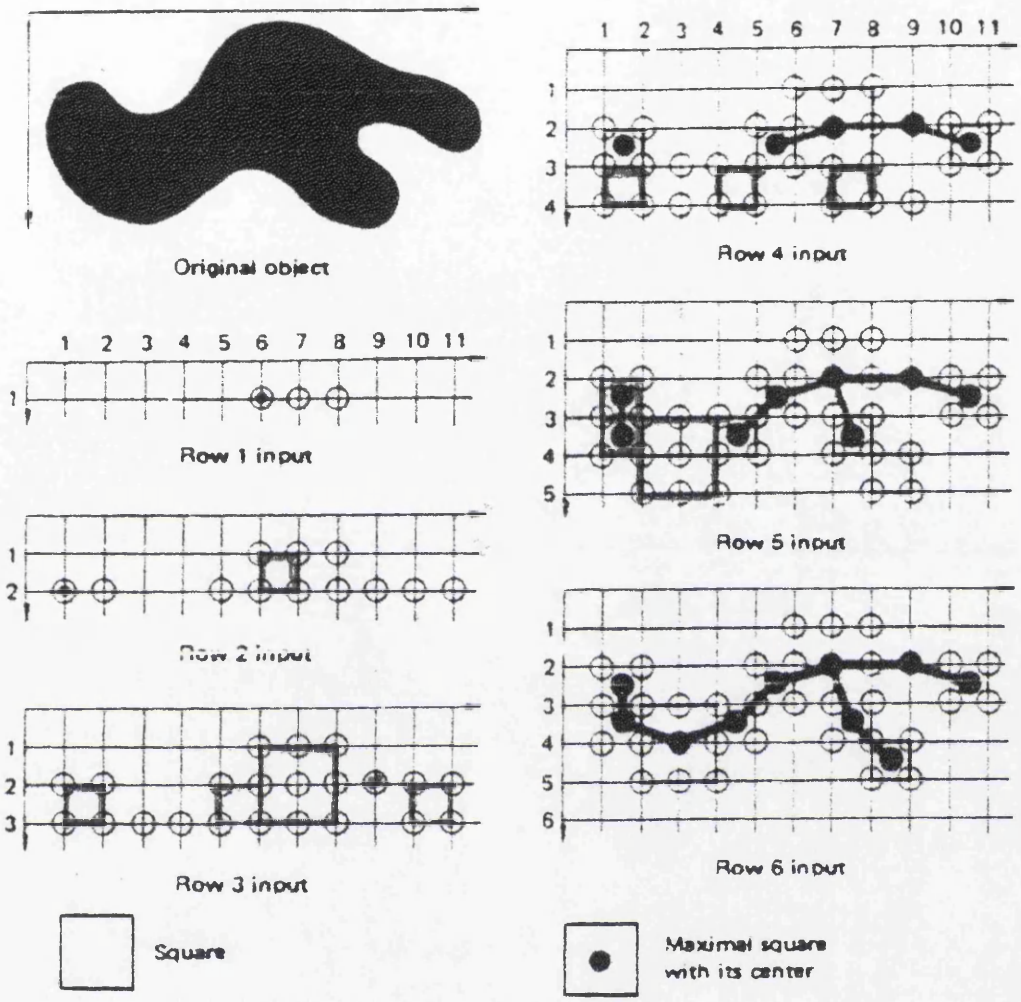

Figure 3.3: MSM

imal square. Once these new squares have reached a maximum their neighbouring pixels will in turn be checked for further squares and so the operation iterates.

In the example illustrated in figure 3.3 the maximal square $((6,1),2)$ yields three bordering squares $((5,2),1)$, $((9,2),0)$ and $((7,3),1)$. The former two new squares are also found to be a maximum, from which are derived two more squares $((4,3),1)$ and $((10,2),1)$ respectively. In this way neighbouring squares are connected to each other and their centers are linked to produce the corelines. The complete skeleton of the object is obtained on inputting the sixth and last line.

The centre points of each maximal square can be easily calculated from their ordered pairs. As a maximal square is reached it sets up links with its bordering squares. It is this relationship between neighbouring centre points that defines the skeleton of an object. Like the medial axis transform techniques the original object can be reconstructed from the lengths of each square. But unlike previous methods the resulting skeleton is independent of coordinate values.

To complete the study of thinning algorithms mathematical morphology techniques are also examined [15]. Most image processing is based on calculus and the continuous image, however mathematical morphology is a completely separate to the standard image processing. It is based on geometry and shape. Morphology operations are concerned with preserving the basic shape of an object. Set theory is used in all its operations. Here only binary images will be considered though the principle can be extended to grey scale.

A binary image can be treated as a two dimensional point set. Pixels forming part of the object are given the value one and belong to the set X. All other background pixels are assigned the value zero and form the set X^c. The origin of the image is marked with a diagonal cross and has co-ordinates $(0, 0)$. The remaining pixels are calculated as $(row, column)$ from the origin. Each pixel x belonging to the set X can be treated as a vector with respect to the origin.

Operations are carried out on the object X using a structuring element B. Structuring element B forms another, usually smaller, set. It too has a local origin known as the representative point, which is also marked by a diagonal cross. Set operations are then carried out using both sets X and B. Set B is moved systematically across every pixel in the image. The image pixel which lies under the representative point of B is called the current pixel. The result of the operation between X and B at the current position is stored in that current image pixel.

Two operations fundamental to mathematical morphology are dilation and ero-

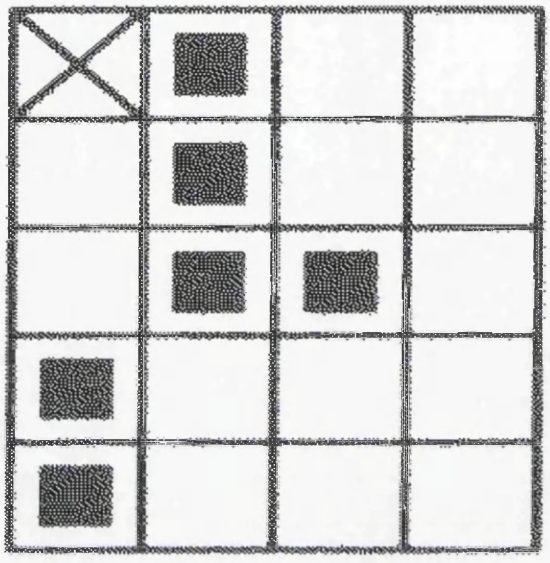

Figure 3.4: Point Set

sion. Dilation $\oplus$ increases the object's size by filling in small holes and gaps. It is sometimes called the fill or grow operation. The dilation $X \oplus B$ is the set of all points formed by the vector addition of each pair of points from the sets X and B.

$$X \oplus B = \{d \in E^2 : d = x + b \text{ for every } x \in X \text{ and } b \in B\}$$

This is illustrated in the following example.

$$\begin{aligned} X &= (0,1),(1,1),(2,1),(2,2),(3,0),(4,0) \\ B &= (0,0),(0,1) \\ X \oplus B &= \{(0,1),(1,1),(2,1),(2,2),(3,0),(4,0), \\ &\quad (0,2),(1,2),(2,2),(2,3),(3,1),(4,1)\} \end{aligned}$$

Erosion $\ominus$ acts to simplify the object and results in decreasing its size. It is also known as the shrink or reduce operation. Erosion strips away isolated pixels

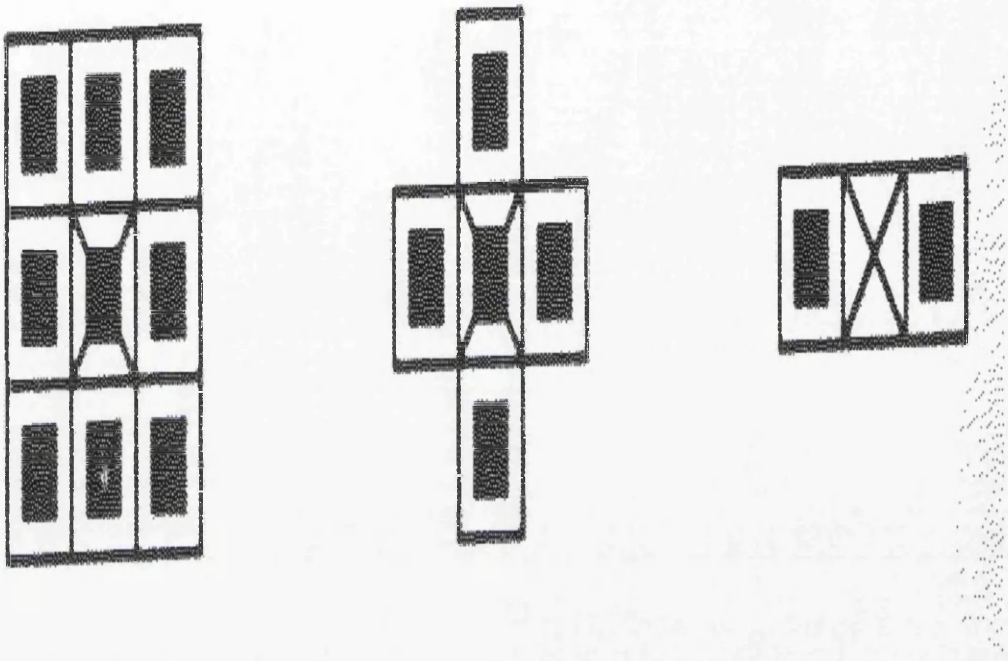

Figure 3.5: Structuring Element

and small parts of the object if they are smaller than the structuring element. The current pixel belongs to the set $X \ominus B$ if all points in the structuring element B are also points in the object X.

$$X \ominus B = \{d \in E^2 : d + b \in X \text{for every } b \in B\}$$

This is illustrated in the following example.

$$\begin{aligned} X &= (0,1),(1,1),(2,1),(3,0),(3,1),(3,2),(3,3),(4,1) \\ B &= (0,0),(0,1) \\ X \ominus B &= (3,0),(3,1),(3,2) \end{aligned}$$

Thinning a line combines both dilation and erosion into one operation called the hit or miss transformation $\otimes$. Instead of having one structuring element B, the hit and miss transformation requires a composite structuring element, where

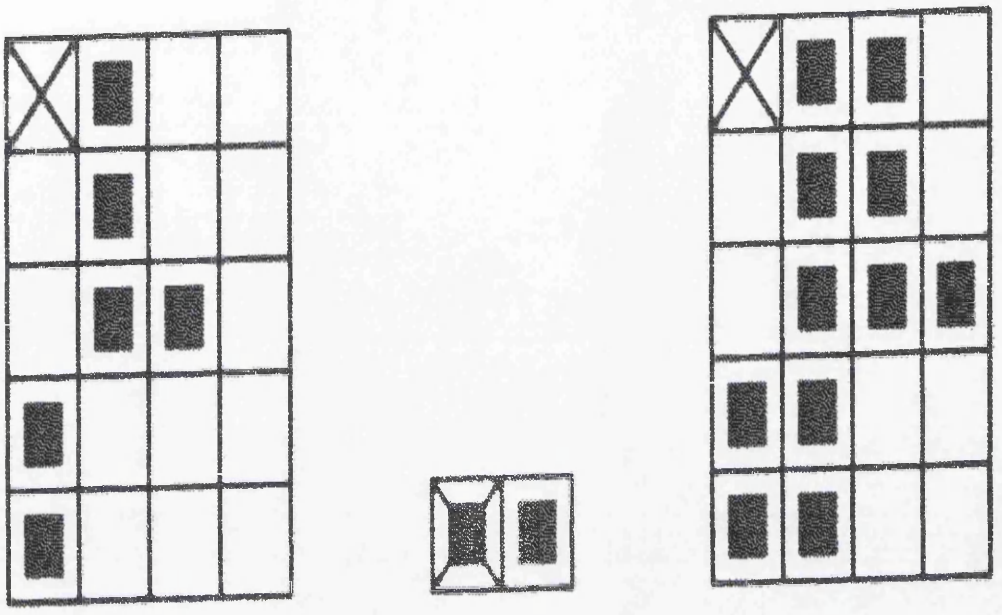

Figure 3.6: Dilation

$B = (B_1, B_2)$ and B_1 and B_2 are disjoint sets. In order for the current pixel to be in the hit and miss transform, B_1 must be fully contained in the set X and all of B_2 should form part of the background as the following formula describes.

$$\begin{aligned} X \otimes B &= \{x : B_1 \subset X \text{and } B_2 \subset X^c\} \\ &= (X \ominus B_1) \cap (X^c \ominus B_2) = (X \ominus B_1) | (X \oplus B_2) \end{aligned}$$

Thinning $\oslash$ is then achieved by stripping away the result of the hit and miss transform from the original line, and is defined as

$$X \oslash B = X | (X \otimes B)$$

Thinning a line often requires more than one composite structuring element and a whole sequence is often the norm. Such sequential thinning can be expressed as follows.

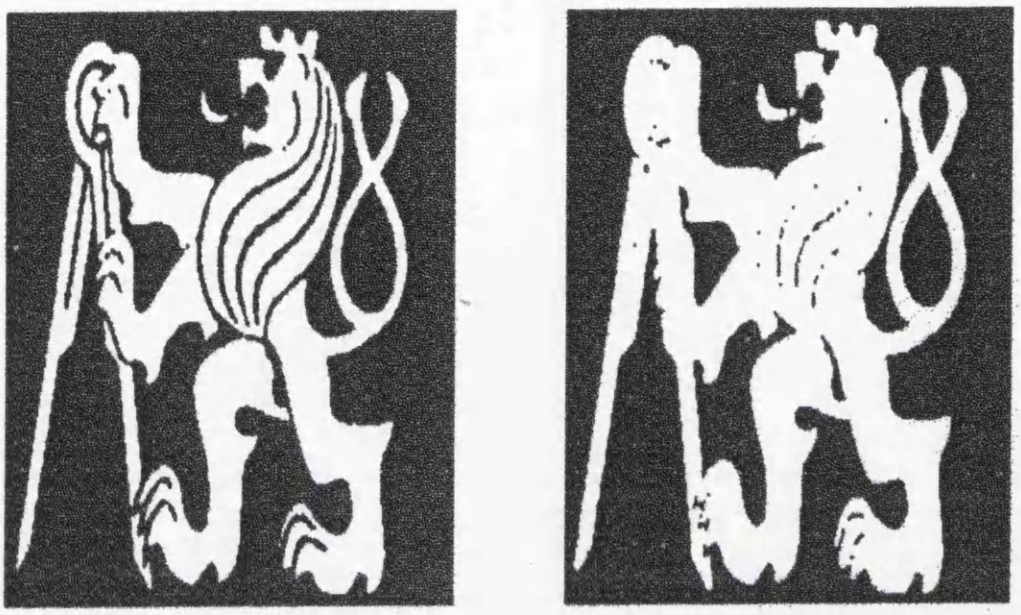

Figure 3.7: An example of Dilation

$$X \oslash B_{(i)} = ((((X \oslash B_{(1)}) \oslash B_{(2)}) \ldots \oslash B_{(i)}) \ldots)$$

A single composite structuring element is usually expressed in matrix form elements belonging to (B_1) are denoted by a 1 while those belonging to (B_2) are set the value 0, a * represents elements of the matrix which belong to neither set and are irrelevant to the calculations. A standard sequence of thinning matrices is as follows.

$$B_{(1)} = \begin{bmatrix} 000 \\ 1* \\ 111 \end{bmatrix}$$

$$B_{(2)} = \begin{bmatrix} *0* \\ 110 \\ 11* \end{bmatrix}$$

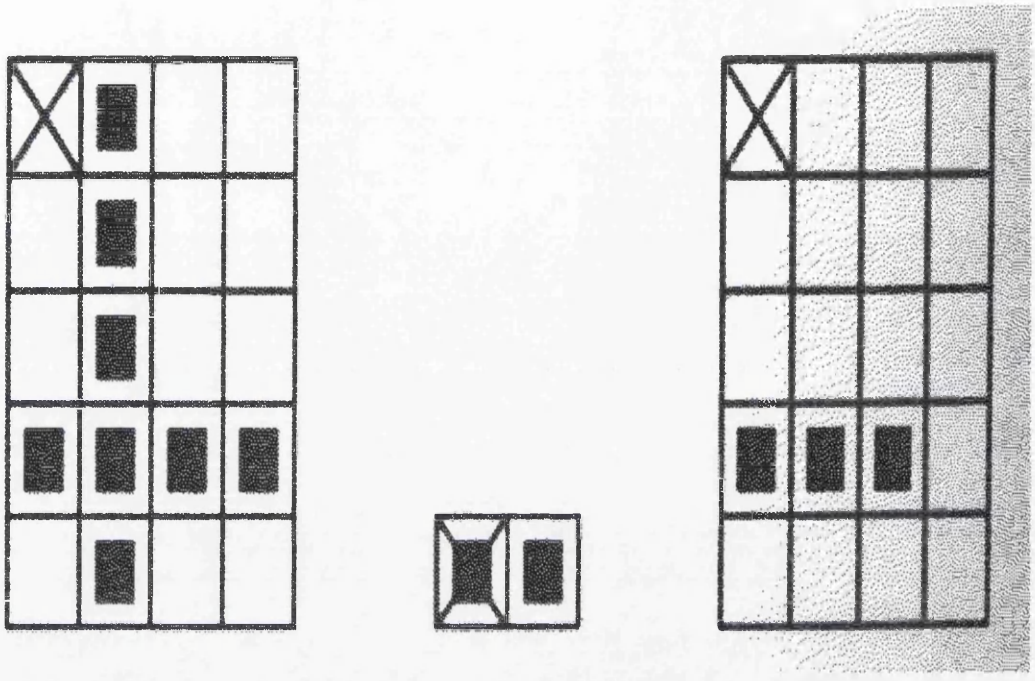

Figure 3.8: Erosion

$$B_{(3)} = \begin{bmatrix} 000 \\ 1* \\ 111 \end{bmatrix}$$

$$B_{(4)} = \begin{bmatrix} 000 \\ 1* \\ 111 \end{bmatrix}$$

Although achieved by very different means, the thinned lines resulting from the mathematical morphology technique bear quite a few similarities to all other thinning methods expect for the maximal square moving algorithm. In each, the outer edges are repeatedly stripped away until what remains is a list of connected pixels representing the skeleton of the object. Thus a raster object is converted into raster lines running through the exact centre of each of the object's lines.

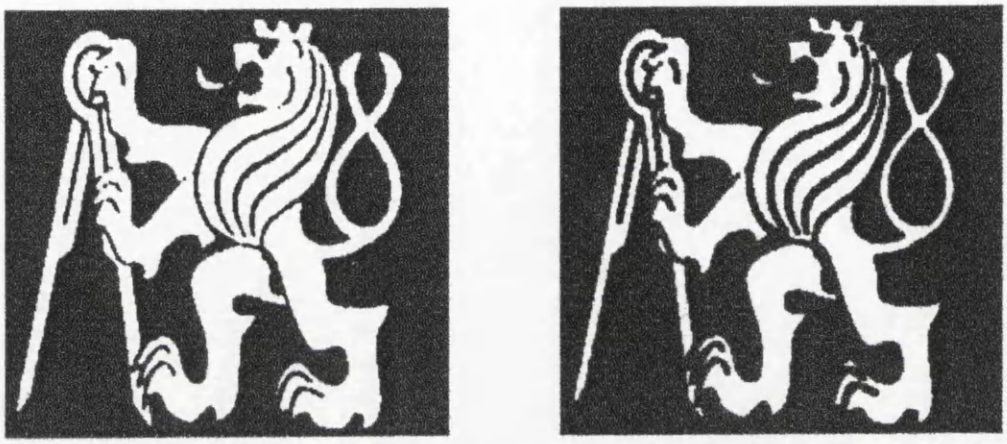

Figure 10.8 *Erosion as isotropic shrink.*

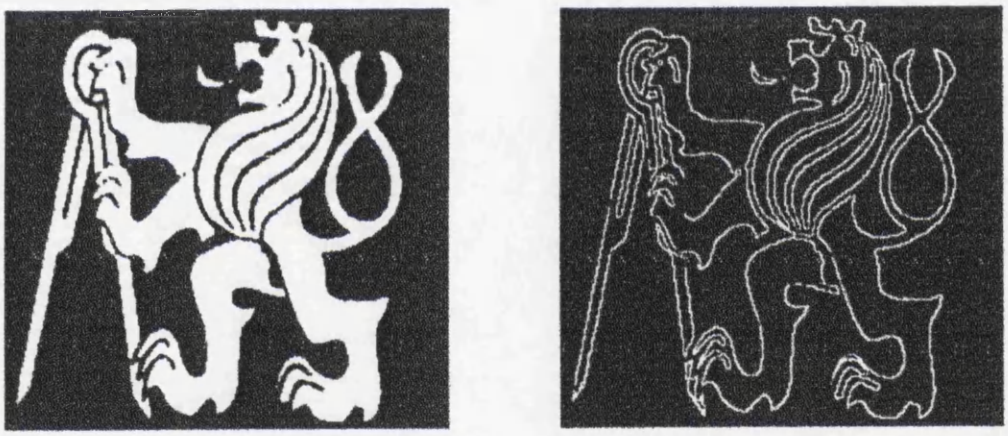

Figure 3.9: An Example of Erosion

The maximal square moving algorithm differs from the rest in that the centre points of the squares are linked together to form the skeleton. It is not guaranteed that these centre points will coincide with actual pixel co-ordinates and 4- or 8-connectivity is not used to connect them together. This freedom of representation enables the centers of the squares to be joined by vectors, combining two desirable properties into one process, the thinning of a line and the vectorisation of it.

Numerous vectorising algorithms already exist [13]. Most have been developed for interactive design such as CAD systems or in the printing of high quality text

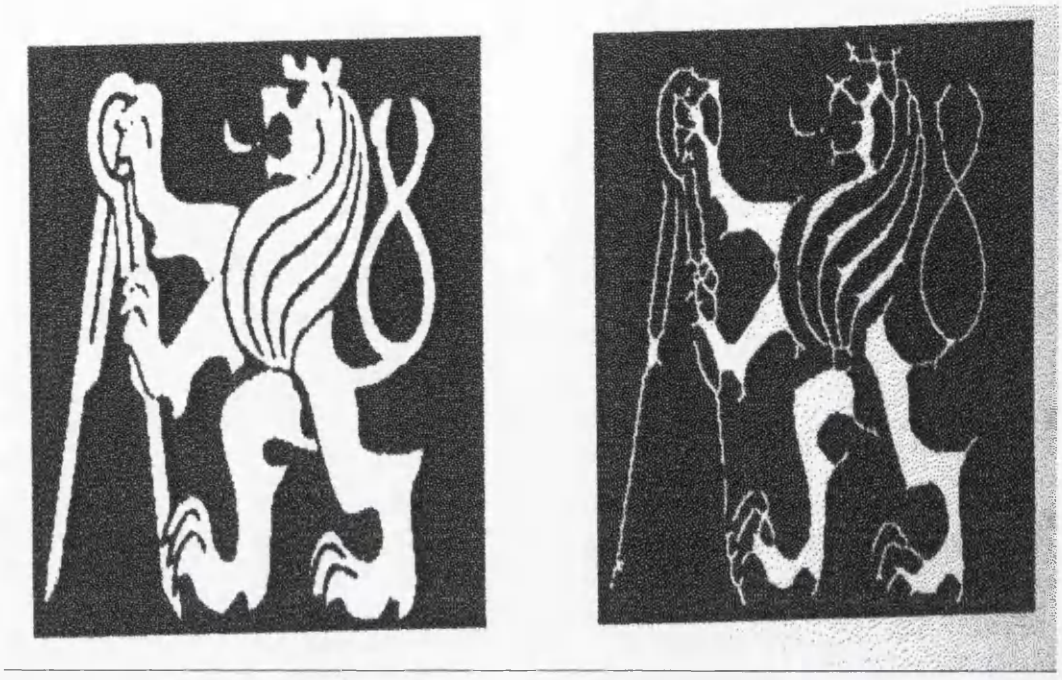

Figure 3.10: Morphology

from scanned in or otherwise digitised fonts. In computer graphics vectorisation is generally known as curve fitting. As its name implies curve fitting is the process of trying to fit a parametric curve representation to a raster image.

Seldom does a single curve equation fit a complete line. Parametric curve representations allow a line to be divided into segments and a curve fitted to each segment. How smooth these curves are and how the segment components fit together depends on the curve representation employed. A wide range of curves, splines and polynomials are available. Each differs in determining the degree of smoothness and continuity that exists at the joints of these curve segments. Just as there are many parametric curve representations, there are many curve fitting algorithms. The algorithm discussed here is that of Bézier curves [17].

A Bézier curve of degree n is defined in terms of Bernstein polynomials.

$$Q(t) = \sum_{i=0}^{n} V_i B_i^n(t), t \in [0,1],$$

where the V_i are the control points and the $B_i^n(t)$ are the Bernstein polynomials of degree n

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i(1-t)^{n-i}, i = 0, \ldots, n.$$

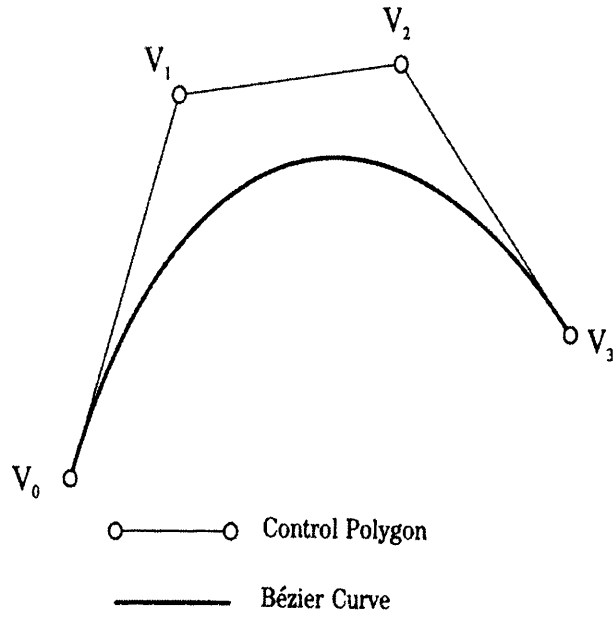

Figure 3.11: Bezier

As has been already stated, a line is usually composed of several curves joined together. In order for the curves to run smoothly into each other there must be some degree of continuity at the joints, a joint being the shared point between two curves where one ends and the next begins. For the basic degree of continuity the second last control point and the last one must be collinear with the first and second control points of the next curve, where the last and first control points are the same.

If the distance between the control point on the left and the shared point is the same as the distance between the shared point and the next control point on the right then the tangent vectors will be equal in magnitude and direction. This is known as parametric continuity and denoted by C^1. However for many application

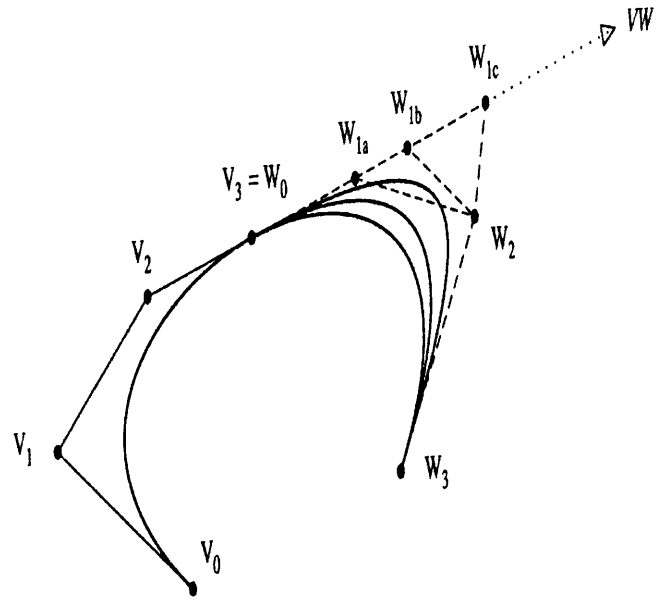

Figure 3.12: Continuity

this is too restrictive and it is sufficient that the tangents be only equal in direction. This is known as geometric continuity denoted G^1 and implies that points need only be collinear and not also equidistant, as the distances do not effect the smoothness at the joints.

This extra degree of freedom allows the algorithm employ a least-squares fitting method which sets the distances between knots so that the error between the digitised points of the fitted curve is minimised. This has the advantage that fewer segments are needed to fit the curve.

The general problem is stated as, given a set of points P in the plane, find a curve Q that fits those points to within some given tolerance. In this case the curve is a Berstein-Bézier curve and the fitting criterion is to minimise the sum of the squared distances from the points on the original line to their corresponding points on the curve.

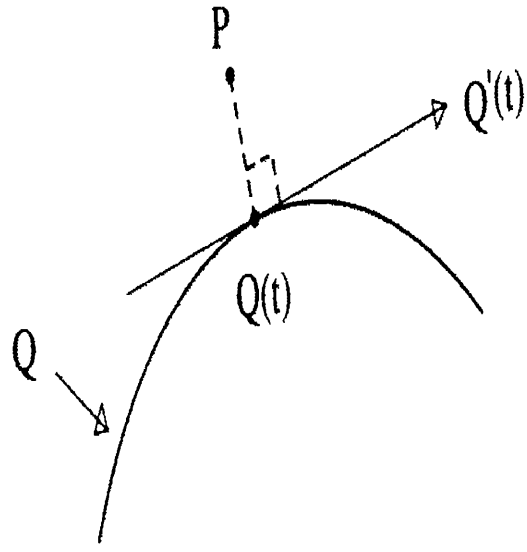

Figure 3.13: P-to-Q(t)

The problem is finding for each point its corresponding point on the curve. Or stated another way is finding the parametric value t associated with each point P. A traditional approach to estimating the value of t for each digitised point is to use chord-length parameterisation.

Once each point has an associated t value, we can fit a cubic Bézier segment to the set of points and compute the error by comparing the distances between each digitised point p_k and the point with parametric value t_k on the generated curve. The square distance between a given point P and a point $Q(t)$ on a parametric curve Q is

$$dist = \| P - Q(t) \| \; or [Q(t) - P] \cdot Q'(t) = 0$$

The first step is to compute approximate tangents at the endpoints of the digitised curve. There are several options to achieving this, such as fitting a least-squares line to the points in the neighbourhood of the endpoints or by averaging vectors from the endpoints to the next n points.

Next an initial parameter t is assigned to each point on the digitised curve using chord-length parameterisation. The first and last control points are then fitted to the endpoints of the current section of the digitised curve. The two inner control points are placed a distance α_1 and α_2 away from the first and last control points in the direction of the unit tangent vectors previously computed.

Then the error between the Bézier curve and the digitised points is computed. If the fit is not acceptable the digitised points are broken into two subsets at the point of greatest error. The unit tangent vector is then calculated at the point of split and recursively new Bézier curves are attempted to be fitted to the two new subcurves. This process continues recursively subdividing until the fit is accepted.

However the initial chord-length parameterisation of the points was only a rough approximation. If there was a better parameterisation of the points there may be a better fit of the curve without further recursive processing. Fortunately the Newton-Raphson iteration calculates a better t using the following formula,

$$t \leftarrow t - \frac{f(t)}{f'(t)}$$

Remembering that the least square distance is

$$[Q(t) - P] \cdot Q'(t) = 0$$

the formula reduces for each point to

$$\frac{Q_1(t) \cdot Q_2(t)}{[Q_1(t) \cdot Q_2(t)]'}$$

This technique can greatly improve the fit of the curve to the points. It is fairly inexpensive and tends to converge rather quickly. Though it is cheap the associated fitting attempts are not. However if the initial fit is very poor it may be best not to attempt the improvement. Additionally since the incremental improvement decreases quickly with each successive iteration there is a limit set on the number of iterations made.

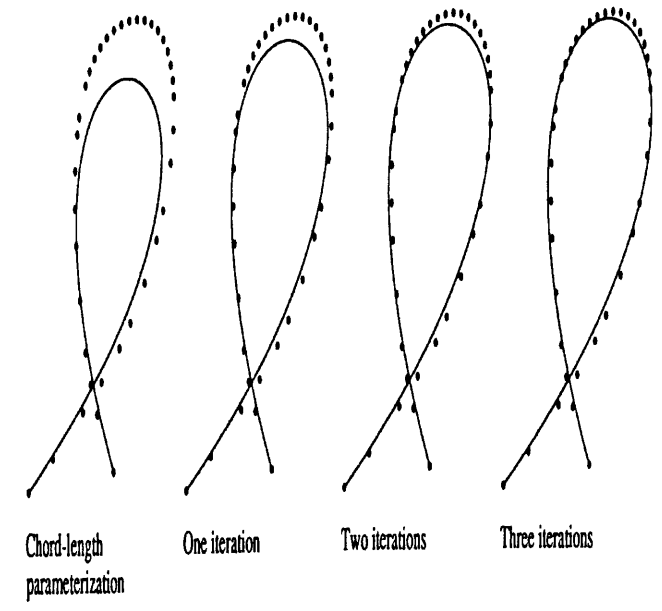

Figure 3.14: Fitting Curve

This curve-fitting algorithm provides one example of how a digitised line can be converted into its vector representation with complete automaticity. However the digitised line must be a clean single line similar to those found in design drawings. This is why CAD system have been able to successfully convert scanned in drawings into their corresponding vector format.

3.3 Proposed Solution.

Lets start by restating the problem: given a sketchy raster image, does there exist an algorithm which can produce the cleaned-up vectorised equivalent? The current solution is to use a thinning algorithm to extract the core-line of the sketch and then to fit a curve to this core-line. The previous section examined the techniques available to solve both the thinning and the vectorisation problems. This section will look at how well these techniques adapt to the problem area.

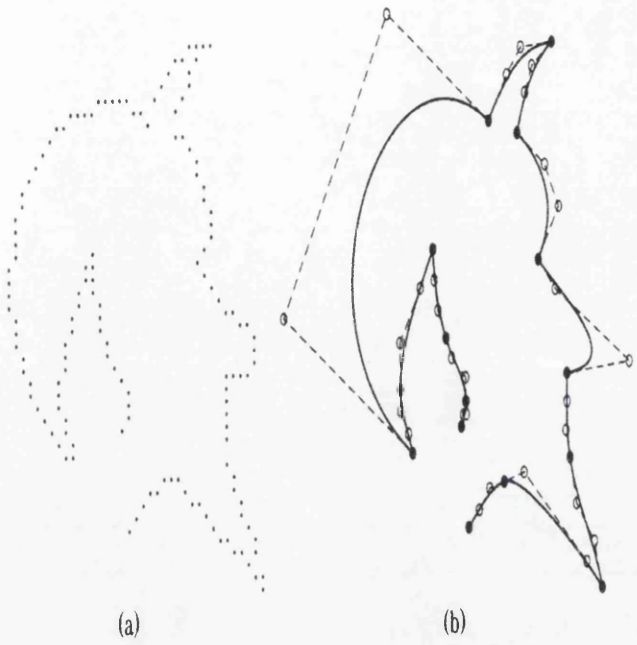

Figure 3.15: Example of a Fitting Curve

First the question of using a thinning algorithm to extract a cleaned-up image from a sketchy image will be discussed.

Most current thinning algorithms run into the following three problems.

- How to extract one line from several sketchy ones.
- The correct position of the line within a group of sketchy ones.
- How to complete vague unfinished lines.

These three problems will now be examined in more detail.

1. A sketchy line is composed of several lines drawn in close proximity. This group of lines represents just one line in the cleaned-up image. Mathematical morphology is the only thinning method which can cope with such a situation. It does so by first filling in the gaps between neighbouring lines, using the dilation operation, thus merging them into one thick line. Thinning is then

performed as usual. All other thinning techniques are very sensitive to the lines border and have no means of dealing with several random sketchy lines.

2. Even if it is possible to merge these sketchy lines into one thicker line on which to operate, the resulting skelton obtained from thinning algorithms is not always in the intended position. Within a sketchy line there exists a darker region through which the desired line runs. If all the lines are merged into one, this dark region does not always lie in the exact middle. However current thinning algorithm work by stripping alway the boundaries of thick lines until all that is left is a single skeleton lie running approximately through the middle of the original line. Each line is treated in the same manner. There is little control over the result. This leads to the incorrect positioning of thinned lines.

3. Most systems have facilities built into them to close small gaps. These deal with very small holes usually caused by noise introduced while scanning. However the gaps occurring in lines produced by vague lines drawn by the artist are often far more substantial than these. Such gaps often prove too large for current gap closing algorithms to resolve. Therefore the line would remain incomplete.

As the curve fitting algorithm depends on one single complete line to which to fit the curve, and thinning algorithms fail to extract such a line, the vectorisation process can not proceed.

The problem with current algorithm is their lack of control over the outcome. Each line is treated in exactly the same manner whether it belongs to a group of sketchy lines or is a vague incomplete one. Obviously both these cases must be dealt with by very different methods. The global nature of thinning algorithms prevents any differentiation being made between line types.

What is needed is some sort of "intelligent" edge detector which can determine what form of line it is dealing with and alter its behaviour accordingly. Such a technique requires inbuilt knowledge of the problem area. Most edge detectors operate solely from the information provided by the image itself and like the thinning

algorithms, treat each image in a global fashion.

Fortunately there does exist an edge detector which has more control over its results than these mere low-level processes. Active control models, more commonly known as snakes, allows information about the problem area to be programmed into it. It can then decipher what type of line it has obtained and deal with it in the appropriate manner. Such built-in facilities are necessary to obtain the correct cleaned lines from the sketchy image.

The next section will study the development of snakes and examine its properties which enable it to cope with vectorising sketchy images.

Chapter 4

Snakes: Theory and Applications:

4.1 An Introduction to Snakes.

Edge detection is one of the most fundamental operations in both computer vision and image processing. It is considered a low-level process as it forms the basis for many high-level functions. Edge detectors generally operate uniformly over the whole image. Only information obtained directly from the image is used in locating the edges.

In order to detect an edge, it is important to understand what an edge is. An edge occurs in an image if there is an abrupt change from a dark region to a bright region or vice versa. To determine whether a pixel belongs to an edge or not it can be compared to its neighbouring pixels. Within a constant region of colour, there will be little if any difference between neighbouring pixels. However when a pixel forms part of an edge, one of its neighbouring pixels will be dark and the others bright so there is a substantial variation in the pixels' values. If the difference between neighbouring pixels is plotted the image gradient is obtained. An edge occurs where the gradient in the image is great, in other words where the difference in neighbouring pixels is also great.

The image gradient is obtained by differentiating the image's intensity values. With a discrete image, a mask is used to calculate the partial differentials on a local scale. A typical edge detector consists of a 3x3 mask which subtracts the neighbouring pixel values from the current pixel in order to obtain the image gradient. Most masks only calculate the edges in a single direction and so four or eight variations of the mask are needed to find all possible edges in the image. The results of each are then combined to extract all the edges [18].

Marr took a slightly different approach to the classical image processing method of abstracting edges [6]. By further differentiating the images gradient he noted that edges occurred where the graph crossed the axis. He called this position the zero-crossing. He stated that by obtaining the second derivative of an image, edges occur when the result is approximately zero [7].

Furthermore Marr suggested that to correctly obtain all the lines in an image the image should first be blurred and then the second derivative calculated. Repeating this process while gradually reducing the blurriness of the image, produces a more accurate result. In order to achieve this he built a Gaussian filter into his mask which has the effect of blurring or smoothing an image. His technique is known as the difference of the Gaussians or DoG and is well used operation in computer vision and image processing.

However many problems occur with these standard operations. The edge detector returns a unique solution stating categorically the exact path of an edge. This result is propagated up through to the high processes. No knowledge of the problem area is incorporated into the detection. Therefore if several possible solutions exists, this low-level process has no means of determining the correct result. If the result is incorrect this error is propagated up to the higher levels with no means of correction. In fact Marr made a strong statement of this view saying that in the process of understanding an image, no higher knowledge is brought to bear until after the relative depths of the objects comprising the image have been calculated. The computations proceed by utilising only what is available in the image itself.

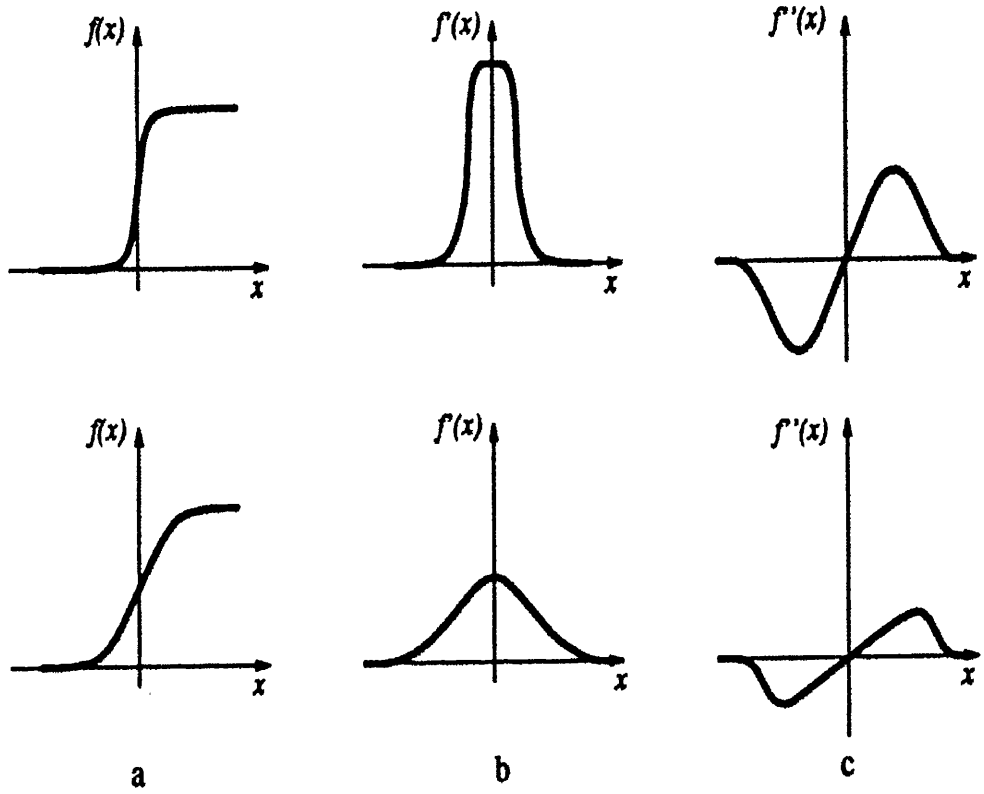

Figure 4.1: Zero-crossings

In the last decade Kass et al. noted that

> this rigidly sequential approach propagates mistakes made at a low level without opportunity for correction. It therefore imposes stringent demands on the reliability of low-level mechanisms [12].

Instead they proposed a technique which identified a range of possible solutions and used prior knowledge to identify the correct one. Such a technique was called active contour models or more commonly referred to as snakes. It was based on the more general technique of deformable models. The next section shall trace the development of snakes.

4.2 The Theory of Snakes.

Kass's snake is not limited to detecting salient edges within an image. Any feature, that can be expressed as a mathematical expression, can be located. The first snake, that Kass proposed sought out lines, edges, terminations and subjective contours [12]. Thus it proves to be more than just a powerful edge detector.

His original snake was a programmable spline. The spline was manually placed near to the feature in the image which the user wished to locate. The snake has been programmed to be attracted to this feature and it moves by its own means the rest of the way until it locks accurately onto the feature.

This attraction to a certain feature can be described as a force pulling the snake towards it. Equally snakes can be programmed to be repelled away from other features. Indeed it is these forces which enable a snake to move as they pull and push the snake into the desired position.

There are three types of forces acting on the snake. The spline itself has its own internal force, this dictates the smoothness and elasticity of the final result. The image of course has forces pulling the spline towards the desired features and possibly pushing it away from inaccurate results. The final force comes from an external source. It enables prior knowledge of the problem area to be available to the snake. Such information can be provided directly by the user or may be passed to it from higher level processes.

Forces are denoted in the snake's equation as energy functions. The stronger the force, the lower its energy value. In this way the snake is constantly trying to reduce its overall energy and therefore is described as an energy minimising spline. If the spline is defined as $v(s) = (x(s), y(s))$, then snake's total energy can be expressed in the following formula.

$$E^*_{snake} = \int_0^i E_{snake}(v(s))ds$$

$$= \int_0^i E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds$$

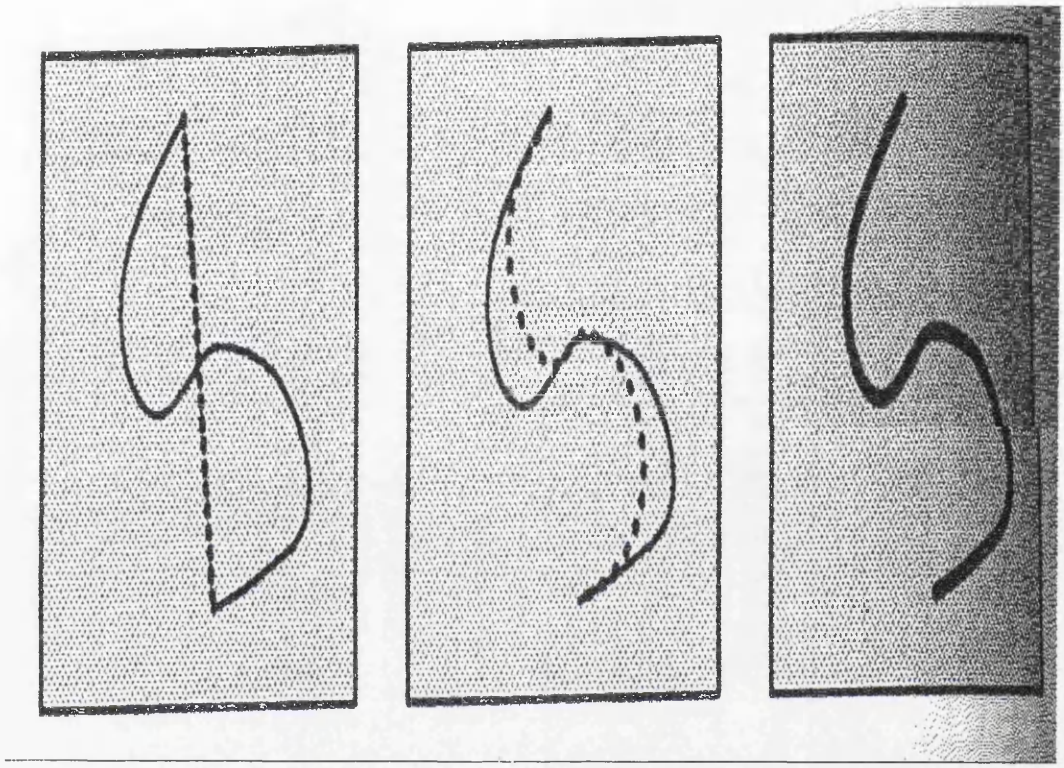

Figure 4.2: Snake

The first component of the equation $E_{int}(v(s))$ represents the snakes own internal energy. Two forces control the internal structure of the spline and are expressed in the following equation.

$$E_{int} = \frac{(\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)}{2}$$

The first part of the equation $\alpha(s)|v_s|^2$ determines the elasticity of the snake. This first-order term allows the snake to expand or contract depending on its sign. It acts like an elastic band, stretching and shrinking to fit accurately onto the desired feature. The elasticity of the spline is controlled by its weight α. If α is set to

zero, a discontinuity is allowed to occur at that point on the snake.

The second part $\beta(s)|v_{ss}(s)|^2$ effects the smoothness of the snake. It behaves like a thin metal plate dictating the extent to which the snake may bend. Like α, β is a weight, it controls the flexibility of the spline. When set to zero it enables a second-order discontinuity to occur, thus forming a corner at that point. By altering the values of α and β the splines shape can be controlled.

The second component of the snake's total energy $E_{image}(v(s))$ describes the forces from within the image acting on the snake. Image features which attract or repel the snake are programmed as energy functional. This energy provides the force necessary to enable the snake to move to its desired location. As long as a feature can be expressed as a mathematical expression its energy functional can be derived. Kass's original image energy came from the lines, edges and terminations within the image and was expressed as follows.

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}$$

The weight associated with each feature determines how much force it exerts over the snake. By adjusting them the snake can be made to act in many different ways.

The simplest energy functional to define is the line, it is just the intensity of the image itself $E_{line} = I(x, y)$. The image intensity reflexes how bright or dark the image is at any point, so depending on the sign of w_{line} the snake can be attracted to bright or dark lines. The exact line the snake locates is dependent on the snake's initial positioning within the image.

The second energy functional locates nearby edges. An edge occurs where there is a sudden significant change in intensity within the image. Such information can

be obtained by examining the image's gradient. A large gradient indicates the presence of an edge at that point. Conventional edge detectors use this technique to determine the location of edges. Many variations of detectors exist and any one of them can be used to provide the edge's energy functional $E_{edge} = -| \bigtriangledown I(x,y)|^2$.

The success of locating of an edge again depends on the snake's initial positioning. If the snake is placed too far away from the edge at the start, the edge's force will be too weak to attract the snake towards it. One means of increasing the force's range is to blur the image. When an edge is blurred, it is extended over the neighbouring pixels, thus increasing the width of the edge and hence the force's catchment area. Once the snake has been attracted to the area of lower energy, the blurriness is gradually reduced. As the blurriness is reduced the snake can adjust itself to lock more accurately onto the edge.

Marr-Hildreth also incorporated this notion of scale space in their edge detector. So Kass adopted this edge detector as their edge functional ($E_{edge} = -(G_\sigma * \bigtriangledown^2 I(x,y))^2$, where G_σ is the Gaussian filter which blurs the image and σ controls the degree to which it is blurred.

The third energy functional is the most complicated of the three, it locates the corners or termination of lines within the image. To do this the image is first smoothed slightly to form image g. The gradient direction along the spline is denoted by $\psi(x,y)$ and unit vectors along and perpendicular to this gradient direction are defined as $n(x,y) = (\cos\psi(x,y), \sin\psi(x,y))$ and $n_\perp(x,y) = (-\sin\psi(x,y), \cos\psi(x,y))$ respectively. Then the corners and terminations of line segments can be calculated by examining the curvature of the contour as described in the following equation.

$$\begin{aligned} E_{term} &= \frac{\delta\psi}{\delta n_\perp} = \frac{\delta^2 g//deltan_\perp^2}{\delta g/\delta n} \\ &= \frac{(\delta^2 g/\delta y^2)(\delta g/\delta x)^2 - 2(\delta^2 g/\delta y \delta x)(\delta g/\delta x)(\delta g/\delta y) + (\delta^2 g/\delta x^2)(\delta g/\delta y)^2}{((\delta g/\delta x)^2 + (\delta g/\delta y)^2)^{3/2}} \end{aligned}$$

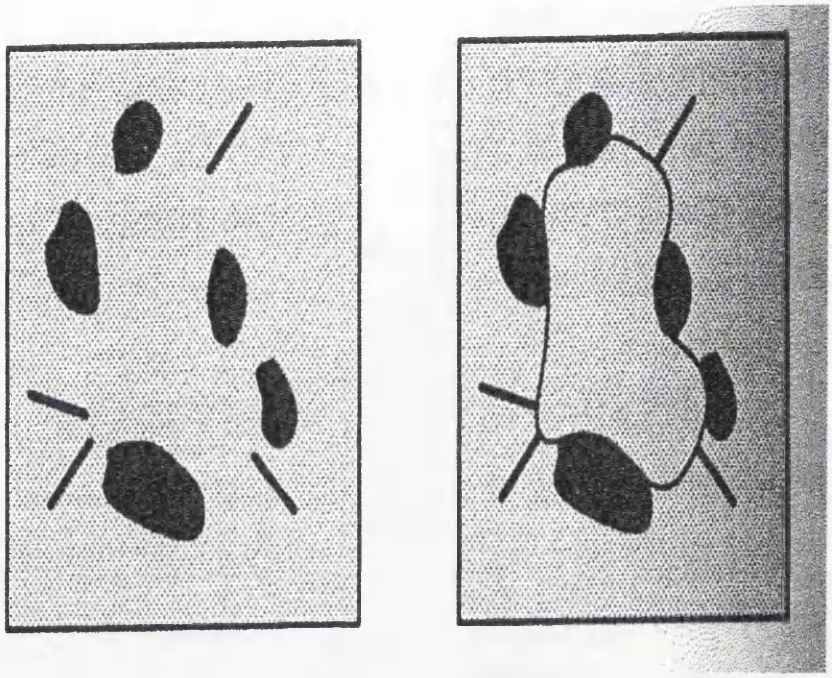

Figure 4.3: Subject-contours

The third component of the snake's total energy equation $E_{con}(v(s)$ refers to any external constraints that are imposed on the snake. These external constraints can come in many forms. Users can interact with the snake forcing it towards or away from features in the image. Alternatively information can be passed to the snake from higher processes or simply prior knowledge programmed into the snake. These external constraints can place large amounts of energy on features deemed undesirable as well as further reducing the energy functional in the presence of desired features.

External energy often provides the vital information in choosing the correct solution when many possibilities exist. In the following example a snake is trying to trace two adjacent rings in a piece of wood. As the rings are very close together it is often unclear the exact path each takes. Figure 4.4 illustrates three possible solutions, the paths intersect, the paths merge into one or the paths run parallel to each other.

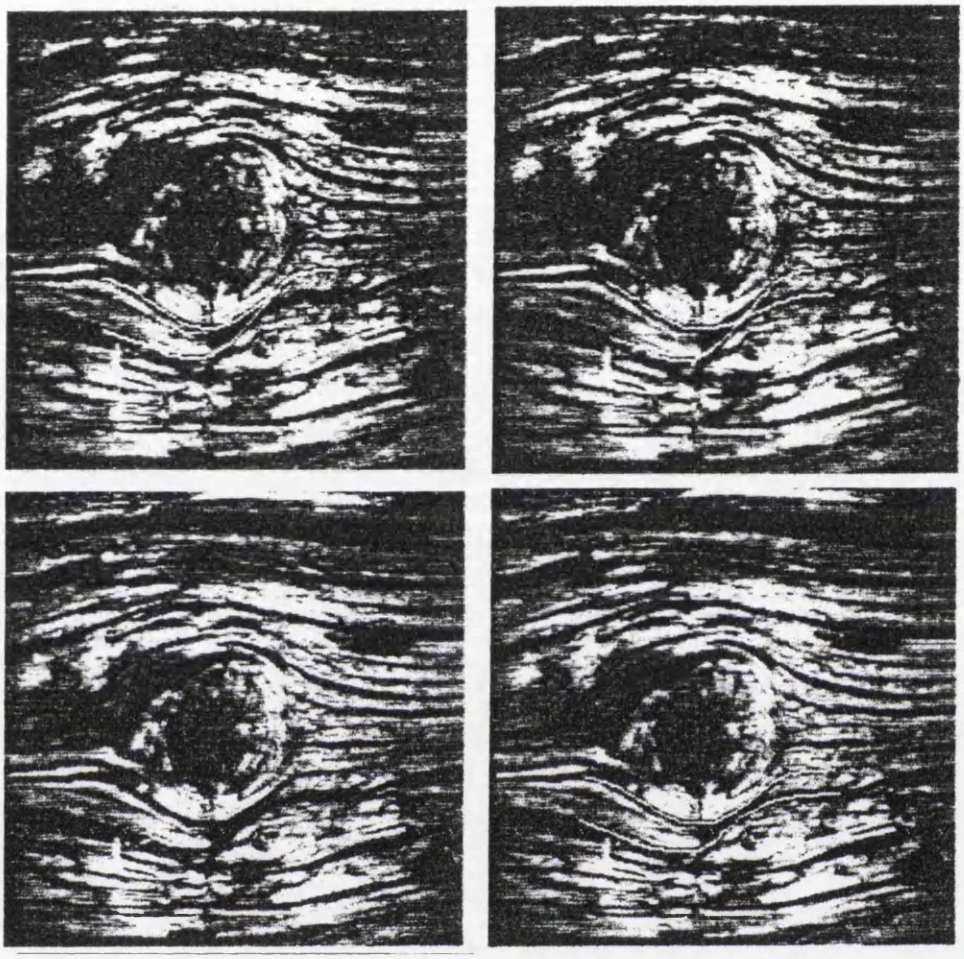

Figure 4.4: Tree-rings

Conventional edge detecting methods have no means of determining the correct outcome. Once a certain path is chosen it can not be changed and it is propagated unchallenged to higher processes. The result can be disastrous. Fortunately a snake enables prior knowledge of the problem area, to be programmed into the snake. Knowing that tree rings must run parallel to each other, enables the snake to discard the former two solution and chose the correct result.

Kass demonstrated the potential of snakes by applying it to two areas of computer vision. The first was the problem of tracking an object. Snakes are well adapted to tracing moving objects, such as speaking lips captured on video. User

intervention or some higher process is needed to place the initial snake in the vicinity of the lips. The snake can then move the rest of the way and accurately locks onto the lips. Interframe speeds in videos are usually fast enough to prevent the object being tracked from altering significantly between consecutive frames. This implies that if the resulting snake from the first frame is superimposed on the next frame, only slight adjustments are needed for the snake to locate the object's new position. Tracking continues in this way, using the snake from the previous frame as its initial position in the current frame.

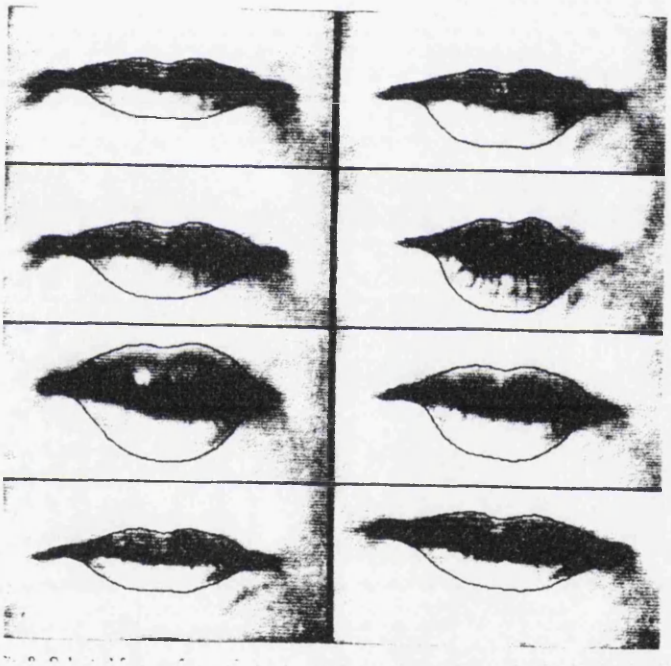

Figure 4.5: Lips

The system can be made more robust by adding interframe constraints. An example of such a constraint would be to assign mass to the snake. Adding mass enables the snake's velocity between frames to be calculated. Once its velocity has been established, the tracked object's future positions can be predicted more accurately.

The second use of the snake was in stereo matching. Our visual system is able to

calculate depth information from a scene because the left eye sees a slightly different image from that of the right eye. Not only is the view different but the angle from which it is seen changes with each eye. The two images produced by the separate eyes are known as stereo pairs. In order to extract the relative depths, the visual system must first locate the same object in both images and then resolve the differences between both views.

Computers can also calculate the relative depth of a scene if the same object can be located in both stereo images. Although the shift in position and angle is only slight, it can be sufficient to prevent convention algorithms from making a match. Snakes resolve this problem in a very different way. Once a snake has accurately locked onto an object in one image it can be superimposed on its pair. The disparity between images is small enough to enable the snake find the object's new position and shape. Having resolved the match, the calculation can begin.

Kass's initial snake has several flaws in it [1]. Its main problem is with the type of mathematics used. Variational calculus is a continuous maths form and therefore treats a computer image as a continuous image. As an image is discrete and not continuous, estimates of the high-order derivatives must be made when calculating the snake's new positions. This may lead to the result falling between the image's discrete coordinate system. Therefore each iteration does not guarantee the optimal solution and often the intermediate results make no sense. This unpredictable behaviour resembles a snake moving, hence its more common name.

Another disadvantage with using calculus is it prevents the use of hard constraints. Hard constraints are those constraints that must under all circumstances be applied. Forces exerted on the snake from the energy functionals are considered to be soft constraints, as the degree to which they are implemented depends on their respective weights and the conflicting forces. Applying hard constraints ensures that each new minimum also satisfies that constraint. If no such minimum can be found the process terminates.

Hard constraints could be used to solve another of Kass's snake's problems. Noisy images are found to greatly hinder the snake's progress. A single piece of noise can attract a snake, lock onto it and then pull neighbouring parts of the snake towards it. Large sections of the snake will then bunch together at this one spurious point. A hard constraint controlling the distance between neighbouring points on the snake could ensure equal spacing and so overcome the effect of noise on a snake.

One positive point in regard to Kass's snake, is that it has the ability to move large distances in just one iteration and so converges quickly on a solution.

Amini resolved many of Kass's problems by developing the snake using discrete dynamic programming. Working on the discrete grid meant that the solution was more stable and each iteration produced an optimal solution with a minimal energy snake. This lead to a more predictable snake. It also ensured that the snake would halt after a finite number of iterations.

Williams further developed Amini's work, maintaining the discrete dynamic programming that allowed for numerical stability and the use of hard constraints [5]. However Amini's algorithm was considered to require a large amount of memory and was very slow. It was also noted that neither Kass nor Amini suggested a method of varying the weights α and *beta* which controlled the internal shape of the snake, as described in the internal energy functional.

$$E_{int} = \frac{(\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)}{2}$$

Maintaining global values for these weights along the whole snake implies β is never set to zero, thus preventing corners from occurring.

Both Kass and Amini made the assumption that the points along the snake were spaced at unit intervals. Based on this, they approximated the derivatives in the internal energy functional to being the distance between two consecutive points as follows.

$$|\frac{dv_i}{ds}|^2 \approx |v_i - v_{i-1}|^2 = |x_i - x_{i-1}|^2 + |y_i - y_{i-1}|^2$$

However when the points are unevenly spaced, the first order term will be incorrect by a factor proportional to the distance between two points. So the greater the distance between neighbouring points the greater their energy values. This has the effect of minimising the distance between points and so causing the snake to contract. Rather than seeking a minimum value Williams proposed that the ideal distance between points should be the average. This encourages even spacing along the spline and helps prevent shrinkage.

With almost even spacing a reasonable estimate of the curvature of the second-order term can be obtained. In addition the curvature of each point is examined after each iteration. If the value is larger than a pre-set threshold, β, the weight of the second order term, is set to zero for that point in the next iteration, allowing a corner to occur. This only happens if the image gradient is above a pre-set minimum, ensuring that corners only occur when they are near an edge.

Berger also noted several flaws in Kass's original energy functional E [2]

- The energy functional contains many unknown parameters, such as the weights of each of the energy functionals and no method is provided to calculate such values. Yet these parameters greatly effect the overall behaviour of the snake.
- The algorithm is not numerically stable.
- Whether a snake can accurately locate a specific feature depends greatly on the initial positioning of the snake.

Berger points out that a problem is well posed when its solution is unique and it depends continuously on the initial data. This is not the case with Kass's snake. Where several contours exist in the vicinity of the initial snake S_0 it is possible that at least two of these will exert the same force E on S_0. Therefore the solution is not unique.

Another problem occurs if the initial snake S_0 cuts several edges or if it lies close to a junction, each of these contours will exert a force on S_0. In this situation the snake will most likely settle in a position between these conflicting forces. Therefore

the resulting curve C does not lie in an actual local minimum and so the snake will not locate any of the contours. The effect of several contours on a single snake can also be that the final curve C only partly lies along a true contour.

Therefore it is necessary to be able to estimate how successful the snake has been in locking onto a true contour. To do this the general definition of an edge is used. An edge is a curve C whose points have a gradient magnitude that is maximal in the direction normal to the curve. Points along the curve then satisfy

$$\frac{\delta|\bigtriangledown I(v(s))|}{\delta n(v(s))} = 0$$

where $n(v(s))$is the normal vector at the point having the arc length s.

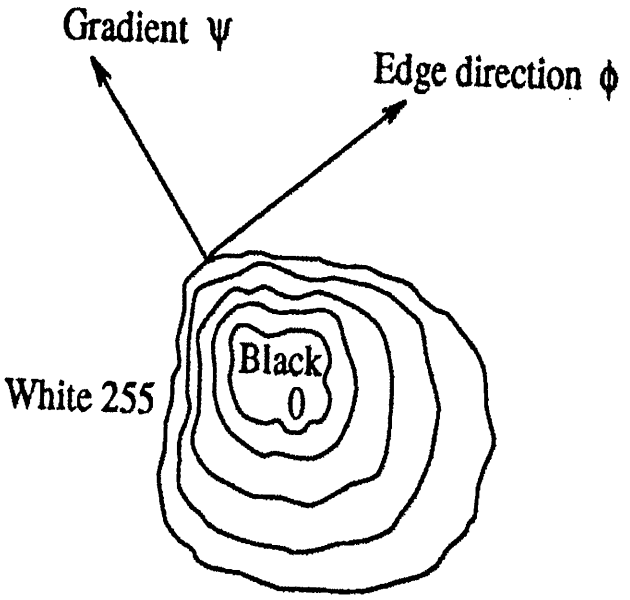

Figure 4.6: Edge Definition

A tolerance level is set to determine which sections of the curve C actually form part of the edge. This set of smaller curves C_i is extracted from C and considered as contours.

Difficulties also arise when deciding the values of certain parameters. A single global parameter γ is used to determine the speed at which a snake converges on a contour. Initially when the snake is still some distance from the contour, γ's value should be small to enable the snake to move quickly. As the snake approaches the contour, γ's value should increase thus slowing down the rate of convergence.

This method works well when the initial snake S_0 is evenly placed away from the contour. However this is often not the case. It is common to find that parts of S_0 lie close to the contour and the remainder is some distance away. As γ is a global parameter with one value for the whole of the S_0, there exist no criteria for determining γ's value.

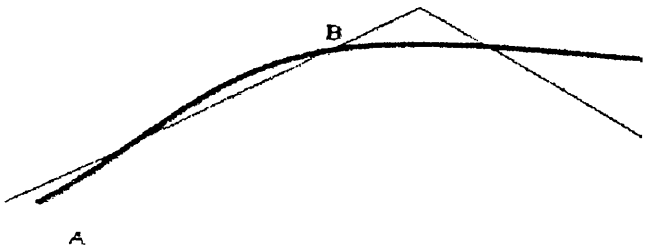

Figure 4.7: Gamma

A solution is to subdivide the initial snake S_0 into several sections depending on their distance from the contour. Each section can then behave as an independent snake. A different γ value can then be calculated for each section.

As all the numerical and convergence problems described previously emanate from the initial positioning of the snake S_0, it seems paramount that this is done correctly. Berger identified the following three conditions for initialisation that should generally be satisfied.

- S_0 must cut the contour in at least one point, because the convergence process will start at these intersections points.
- S_0 must have a direction close enough to the contour which will be detected.
- S_0 must lie in the influence area of at most one contour.

In fact Berger developed a technique which automates the initialisation of the snake S_0 [3]. From any initial snake S_0 position, the number of intersections it makes with contours can be determined by examining the gradient curve along S_0. This set of intersection points (I_j) is used to cut S_0 into curves S_j. Each new curve now contains one intersection point with a contour, thus satisfying at least one of the initialisation conditions.

With each of the problems discussed so far, the solution has been to subdivide the snake into smaller section, each section then acting independently of the others. Instead of one snake behaving as a whole to locate a contour there now exists several shorter versions attempting to achieve this goal. In order to do this Berger developed a technique called snake growing.

Snake growing starts with an initial snake S_0 which usually lies in the vicinity of the contour and is short in length. This ensures that the iterate process will converge quickly, producing a curve C_0. There is no guarantee that C_0 lies entirely along a contour. The estimation process extracts $Cont_0$ from C_0, where $Cont_0$ is a true edge. $Cont_0$ is then extended at its extremities in the direction of the tangents to produce S_1. The whole process is now repeated this time using S_1 instead of S_0.

The growth of the length can be a given value l or can be an adaptive value inferred from the previous results. For instance, when the convergence is fast at a

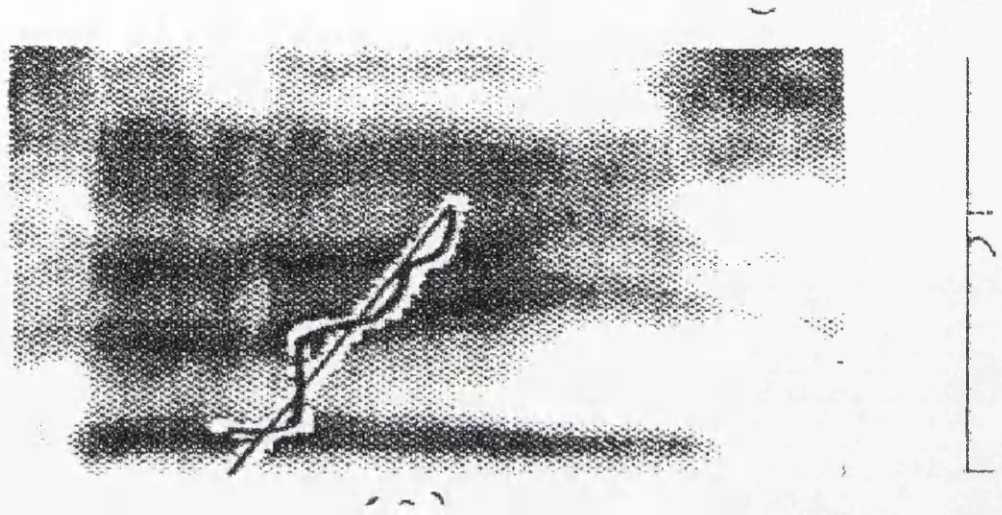

Figure 4.8: Cutting Snake

stage we can suppose that the tangent direction is close to the contour direction; l is then increased.

A whole sequence $S_1, \ldots, S_(n)$ of snakes produces a sequence of contours $Cont_1, \ldots, Cont_n$ whose lengths are increasing. This process is repeated until the snake can't be lengthened any more.

The snake growing algorithm can be described as follows.

- Start from a curve S_0 which is close to the desired contour.
- Run the snake algorithm with S_0 as the initialisation to establish the resulting curve C_0.

- Extract from C_0 the true contour $Cont_0$.
- While lengthening is possible repeat the following.
  - Lengthen $Cont_i$ to produce S_{i+1}.
  - Run the snake algorithm with S_{i+1} to establish C_{i+1}.
  - Extract from C_{i+1} the contour $Cont_{i+1}$.

If the snake growing algorithm fails there are two possible reasons.

1. The edge terminates or there is a hole in the edge.
2. There is sudden high curvature variation in the contour such as the presence of a contour. So the tangent directions at the extremities are far from the contour.

To determine if the contour continues around a corner or after a gap, the extremities of the contour are extended in several directions. Only those solution which provide the best results are retained, or in the case of a junction several solutions may be maintained.

The snake growing algorithm has many advantages over William's more traditional approach. The only time when initialisation is far from the contour is at the beginning of the process. After this the snake is quite near a unique contour with the possible exception of the extremities. Each step guarantees that the snake converges satisfactorily on a contour. Therefore the numerical solution is stable. A disadvantage with this approach is its cost. However as this algorithm is more stable, fewer iterations are necessary.

Whereas the traditional snakes take a more global approach to the locating contours, snake growing operates on a local basis. This enables local behaviour such as corners or high curvature variation to be detected within the contour.

There are as many variations on the snake as there are application areas. Two large areas of research have been robotics and medicine. Robotics is interested in

computer vision and tracking objects. On the other hand medicine usually concentrates on accurately locating features from a scan. Templates of the desired features are often used to initiate the snake and if a sequence of scans is available it is possible to build a three dimension model of the moving organ. Although much of this research is field specific, the techniques employed can easily be adapted for use in other areas, many of which may be applicable to some of the problems associated with computer-assisted animation

4.3 Proposal to Use Snakes to Solve Problem Area.

Lets recap on the problems the current thinning algorithms incurred while attempting to clean-up a sketchy image.

- How to extract one line from several sketchy ones.
- The position of the correct line within a group of sketchy ones.
- How to complete vague unfinished lines.

Having examined various properties of the snake in the previous section, it is now possible to see how a snake deals with the same set of problems.

1. The first problem is that of extracting a single line from a group of sketchy lines. Our visual system does this automatically for us. Squinting at the sketchy image emphasises the human's ability to resolve this problem. Squinting causes the image to blur merging the sketchy lines into each other. Computer images can also be blurred with the same effect. This is achieved by passing a smoothing filter over the image.

   Kass's original snake formula incorporated a Gaussian filter to blur the image. The reason was to create a bigger energy well with which to attract the snake. The same function can be adapted here for a very different reason. Once all the blurry lines have been merged into one, Berger's automatic snake initialisation algorithm can be employed to locate these blurry lines. This creates the

right number of initial snakes from which the cleaned lines can be determined.

2. The second problem incurred was determining the true position of the intended line. It helps to examine how an artist would resolve this problem. The sketchy lines become more concentrated as the true line is approached. This concentration in drawing forms a darkened region with in the sketch. It is through this dark region an artist would run a smooth line, thus locating the intended line.

   This prior knowledge of the true line's position can be programmed into the snake. Having found the approximate area of all the lines, the blurriness of the image can be gradually reduced. The snake now seeks to lock onto the local dark area. As the image becomes less blurry, the result becomes more accurate. The final position of the snake is taken to be the true position of the intended line.

3. The previous two problem resulted from too many lines representing just one line. The last problem is the complete opposite, too little information is available in an incomplete vague line. Once again lets consider how an artist would resolve this dilemma. Our visual system does it for us by continuing the flow of the line until it meets another line and closes the gap. Therefore lines are only left unfinished if it is obvious to the eye how to complete them.

   Our inbuilt mechanism for completing lines is based on the same principle as Berger's snake growing. To continue the flow of the line, the endpoints are extended in the direction of the tangent, until they meet another endpoint or a line. Sometimes blurring the image resolves this problem at the start, as the blurred lines may appear joined. If the initial snake is closed it will remain so as the snake locks on more accurately.

An additional advantage of using snakes is, once the true lines have been extracted form the sketchy image, the vectorisation process is also complete, as the snake is itself a spline. This previously two stage problem is now reduced to one. The less processes employed, the less the risk of introducing errors into the result. Unfortunately, just as human inbetweeners are prone to mistakes so are computer systems. The next chapter will look at the possible limitations of snakes and propose a means of solving them.

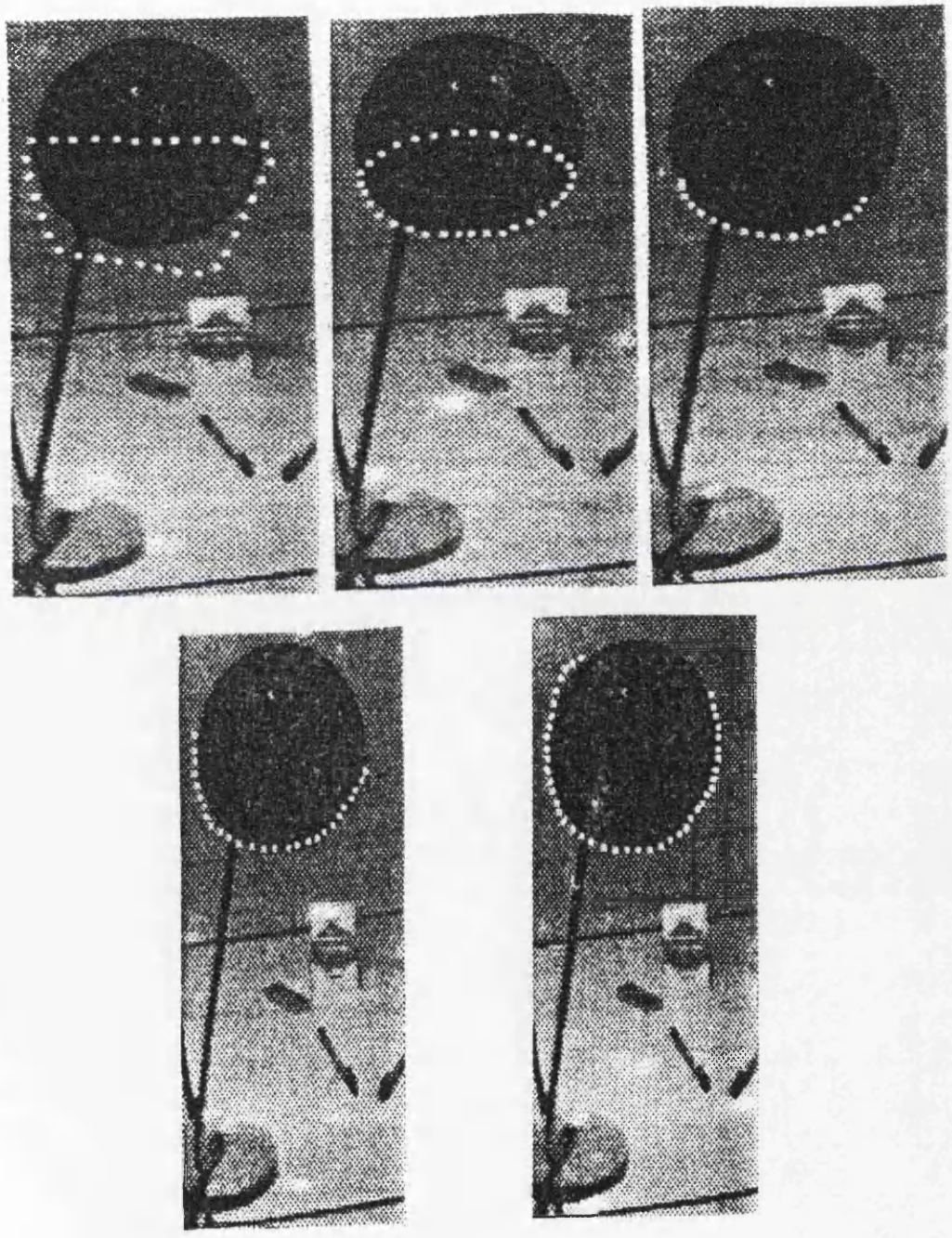

Figure 4.9: Snake Evaluation

Chapter 5

Limitations and Further Work:

5.1 Problems with Proposal.

Errors occur with automating the cleaning up phase for much the same reason as they occur with humans. If the image is unfamiliar, there may not be enough information in the image itself to determine the correct result. In such a situation, where there is no prior knowledge available, both the computer and the human must either seek help or guess at the correct solution. As the computer usually has less prior knowledge at its disposal, it is more susceptible to errors.

Inevitably, vectorising a sketchy raster image will incur problems. The following suggest possible errors which will need further attention.

1. Extracting a single line from a sketchy image. In order to locate a single line, the image must be blurred to blend the sketchy lines into one. But is there one optimal blurriness from which all lines can be determined? In blurring the image, the vague lines may be removed altogether and therefore never located. It may be necessary to allow some human intervention to determine the blurriness of the image. Using a slide ruler a user could set the optimum image from which to initiate the snakes. Often however, one blurry image would not be sufficient to capture all the lines and therefore initiation would have to commence on varying degrees of blurriness.

2. Completing an unfinished vague line. Blurring not only runs the risk of removing vague lines from the image but also in blending separate neighbouring lines together. The animator might have drawn several individual lines which the snake then misinterprets as representing just one. Another possibility is that an unfinished line may be closed but not to the correct line. There must be some easy means of overriding incorrect results. One option would be to display all possible outcomes when the snake has not enough information to determine the right solution and allow the user to decide which one is correct.

It obvious that such incorrect choices may be made but the next two problems are not immediately obvious.

1. Maintaining the thickness of the line. The animator can create different effects by varying the thickness of the lines. It is therefore imperative that the snake is able to recreate the same effect. Unfortunately the vector format of the snake has no thickness information and so each line is limited to the same thickness. This greatly reduces the creativity of the artist and must be overcome at all costs. A suggestion would be to infer from the thickness of the sketch, the thickness of the resulting lines. Heavily sketchy lines are more likely to be thicker than vague incomplete ones.

2. Removing additional lines from the image. When sketching a character, the artist often adds in extra features which will not appear in the final version. Features, such as how a limb lies underneath clothing, not only act as an aid to the animator when drawing the character but also offer vital clues to the inbetweener as to how to interpolate these limbs. However such limbs will not appear in the image, so how does the computer system remove such additional information?

   It is important to note that it is not desirable that such information be removed prior to scanning. Firstly it will require extra overheads in the form of over-cleaning and secondly if such information aids the inbetweener it may

also prove vital in assisting the computer equivalent.

Like the current algorithms used to solve this problem area, the proposed snake technique will require some degree of checking and correcting. This incurs overheads. But unlike the present solutions, snakes possess some knowledge of the problem area and therefore less prone to errors.

In addition, if a sequence of animation is scanned in, the first frame will normally require the most correcting. Once this has been successively vectorised the subsequent frames should be easily converted. This can be achieved in a similar fashion to tracking an object. It is doubtful that the second frame will vary much from the first one. So the resulting snake from the first can be used as the initialisation for the second. Most of the information such as the number of lines, which ones are closed and their respective thickness will change little from frame to frame and so only the most minor tweaking needs to be performed on subsequent frames. This greatly speeds up the process and makes using snakes to cleanup and vectorise a worthwhile operation.

5.2 Further Work.

Humans have an unfair advantage over computer-assisted animation systems when it comes to understanding an image. Not only do they have a vast source of prior knowledge but they also have many aids that they can reference when in doubt. Most computer systems ignore the possibility of providing references or any prior knowledge and yet they expect the computer to perform as well as the human. This is impossible. To ease the difficulty of the computer's task, all additional information should also be made available to the computer.

The human's main source of reference is the model sheet. A database of clean vector poses from the model sheets could greatly minimise the time spent checking and correcting. The user could simply chose the pose most similar to the image in

the first frame. Using this as a template it could lessen many of the limitation from the previous section.

- It would determine the correct number of lines in the image and remove many of the errors introduced in an attempt to automate the process. Blurring could still be used to correctly lock on to the lines but there would no longer be the need to initiate the snakes using varying degrees of blurriness.
- The template would also prove useful when trying to decide which lines are open and more importantly the correct path of incomplete ones.
- If the thickness of the lines is available in the template, it can determine the thickness of the lines in the image.
- The lines in the template reveal which lines are the true lines to appear in the final image. If a line is not in the template it can be considered as extra information provided by the artist. Such information can be saved and stored for future use but does not actually appear in the image e, unless of course the user intervenes and indicates that it is in fact a true line.

Snakes have great potential. The ability to program them with prior information could resolve many of the automatic inbetweening problems. For example its ability to gather enough information from previous frame to enable it to predict the animation in future frames could ease the burden of current inbetween problems. This coupled with a computerised version of the exposure sheet could prove a great aid to the goal of automating the inbetweening process.

However it is my believe that a complete automation of the traditional animation studio is not a desirable goal. For one, I do not believe that we will ever have good enough inbetweening algorithms which can mimic the subtle effects of the human artist. And secondly, I do not see automating the inbetweener's task as benefitting the traditional studio. How else will these junior animator's achieve the skills necessary for them to be promoted into more demanding roles. Therefore my aim would be to automate boring predictable sequences of animation which the computer can successively reproduce while leaving the more creative inbetweening to the junior

animator.

Chapter 6

Summary.

At present there exists three commercial types of computer-assisted animation systems. They are

1. Ink and Paint.
2. Automated Inbetweening.
3. Paperless.

Each system claims to automate various phases of the traditional animation guidelines, as are illustrated here in tabular form. A $\surd$ indicates the phases a system has successfully automated. An x suggests that the system goes some way to achieving this goal but usually requires human intervention. A blank implies that the system makes no provisions for this phase.

Guidelines	*Ink and Paint*	*Automatic Inbetweening*	*Paperless*
Script			x
Storyboard			x
Soundtrack			
Track Breakdown			x
Designs			x
Leica Reel			√
Drawings		x	x
Line Test			√
Cleanup	x	x	x
Trace and Paint	√	√	√
Backgrounds	x	x	x
Checking			√
Final Shoot	√	√	√
Rushes	√	√	√
Dubbing	√	√	√
Answer Print	√	√	√

As can be seen from this table, the paperless system attempts to automate nearly every phase of the studio's guidelines. The reason why so many x's occur is because, though it claims to computerise most phases, it really is just substituting a pencil and paper for a tablet and screen. All the drawings still have to be done manually, be it with a different medium. So, though they can claim to have computerised much, very little has actually been automated.

The next table looks at the benefits these three systems make to the animation process. Again the blanks, xs and √s represent, no, some and definite benefit respectively.

Benefits	*Ink and Paint*	*Automatic Inbetweening*	*Paperless*
Automatic Inbetweening		x	
Checking for Errors			√
Speed up Painting	√	√	√
Removal of Limit of Layers	√	√	√
Procedural Rendering		√	
Images from other Sources	√	√	√
Simplifying Shooting	√	√	√
Resolution Independence		√	√
Reuse of Animation	x	√	√

The last of the comparisons between the three systems is the overheads each produce.

Overheads	*Ink and Paint*	*Automatic Inbetweening*	*Paperless*
Over Cleaning	√	√	
Extra Staff	√	√	
Human Intervention	x	√	x
Storage	√		
Retraining	x	x	√
Reduced Quality		x	x

The last two rows of this table are the most crucial to the animation studio. With regard to the paperless system, there is evidence that some animators cannot draw as creatively with a tablet and screen as with the traditional pencil and paper. In addition, having to retrain all the studio's staff to work with an unfamiliar medium, proves just too great an overhead to be considered any further.

The majority of overheads produced by the automatic Inbetweening system are due to its attempt to completely automate the inbetweening process. The problem lies in the currently available inbetweening algorithms. Each has been developed with a different goal in mind. Some concentrate on complete automicity, whereas

others attempt to produce more complex movements. The following table indicates the strengths of the various methods.

	Totally Automatic	*Non-Linearity*	*Complex Movement*	*No Errors*	*Good Results*
CCLI					
Moving Points		√			
Polar					
2D Blending	√				√
Multi-resolution					√
Union of Circles	√			√	√
Stick Figure		√	√	√	
Star-Skeleton	√			√	√

Inbetweening algorithms are still too limiting in their abilities to be of much benefit to the computerising process. They remain too great a task for this thesis, so this overhead will be ignored.

The remaining overheads could be minimised by developing algorithms that can clean-up a sketchy drawing and vectorise a scanned in image. This thesis focuses on this problem area, which can be summarised as follows; given a sketchy raster image is it possible to extract a cleaned-up vector image?

Sketchy images provide several problems which the cleaning process must resolve.

1. Reduce a group of sketchy lines to a single line.

2. Extract the correct position of this line.

3. Close vague incomplete lines.

The traditional means of extracting a core line is by using a thinning algorithm. Several thinning methods are compared with each other and with the proposed snake solution. The results are in the following table.

Overheads	*Medial Axis Transform*	*Maximal Square Moving*	*Mathematical Morphing*	Snakes
Position of Core Line	Centre	Centre	Depends on Structuring Element	Depends on Energy Functional
Format of Core Line	raster	linked centres	point set	spline
Sensitive	very	very	not particularly	very little
Line Restoration	yes	yes	no	no
Human Intervention	none	none	none	removed using snake growing
Ease of Control	none	none	little by varying structuring element	a lot by controlling the energy functional

Snakes allows the greatest amount of control when extracting the core line form its sketchy representation. None of the other methods have the ability to extract the core line from the darkest region of the group of sketchy representatives or complete vague in a manner similar to how the traditional animator does. This is how the snake resolves the three previously mentioned problems.

1. The snake blurs the image to such a degree that the sketchy lines blend into one. Berger's automatic snake initialisation algorithm is then used to locate all the lines.

2. The blurring of the image is reduced as the snake seeks out the darkest local position.

3. Vague lines are completed using Berger's snake growing technique.

The snake has the additional advantage of being in vector form. However four possible problems occur with this using snakes.

1. Failing to locate the correct number of lines in an image.

2. Closing a line the animator intended to remain open.

3. Uniformity in the thickness of each line.

4. Inability to differentiate between true lines and additional information on the drawing.

All these limitations can be minimised if there exists a vectorised model sheet of the character being cleaned stored in the computer. A pose from this can be used as a template from which to initiate the snakes in the current frame. This can indicate,

1. the correct number of lines in the image.

2. which lines remain open and which are to be completed.

3. the thickness of the line.

4. which are the true lines.

Snakes have been developed to solve many complicated tasks in all areas of computer vision, most notably medicine and robotics. If some of these techniques are introduced into computer-assisted animation, the system may become more robust and reliable. The potential of snakes in problem solving is great.

Bibliography

[1] S.Tehrani A.Amini and T.E.Weymouth. Using dynamic programming for minimizing the energyof active contours in the presence of hard constraints. In *2nd International Conference on Computer Vision*, pages 95–99, August 1988.

[2] M-O Berger. Snake growing. In *1st European Conference on Computer Vision*, pages 23–27, August 1988.

[3] M-O Berger and R.Mohr. Towards autonomy in active contour models. In *10th International Conference on Pattern Recognition*, pages 847–851, August 1990.

[4] B.Thomas. *Disney's Art of Animation.* Hyperion, 1997.

[5] D.J.Williams and M.Shah. A fast algorithm for active contours and curvature estimation. *Computer vision, graphics and image processing: Image understanding*, 55(1):14–26, January 1992.

[6] D.Marr. *Vision.* Freeman, 1982.

[7] D.Marr and E.Hildreth. A theory of edge detection. In *Proc. Roy. Soc. (London)*, pages 187–217, August 1980.

[8] E.Catmul. The problems of computer-assisted animation. In *Siggrapgh Computer Graphics*, pages 348–353, August 1978.

[9] E.Goldstein and C.Gotsman. Polygon morphing using a multiresolution representation. In *Computer Interface '95*, pages 247–254, May 1995.

[10] Thierry.Galas J-D. Fekete. E.Bizouarn, E.Cournaire and F.Taillefer. Tictactoon: A paperless system for proffessional 2d animation. In *Siggrapgh Computer Graphics*, pages 348–353, August 1995.

[11] A.Stork J.Madeira and M.H.Groß. An approach to computer-supported cartooning. In A.S.Glassner, editor, *The Visual Computer.* Springer-Verlag, San Diego, USA, 1996.

[12] A.Witkin M.Kass and D.Terzopoulos. Snakes: Active contour models. In *International Journal of Computer Vision*, pages 321–331, August 1987.

[13] M.Plass and M.Stone. Curve-fitting with piecewise paramtric cubics. In *Siggrapgh Computer Graphics*, pages 229–239, July 1983.

[14] M.Shapira and A.Rappoport. The problems of computer-assisted animation. In *IEEE Computer Graphics and Applications*, pages 44–50, March 1995.

[15] V.Hlavac M.Sonka and R.Boyle. *Image Processing, Analysis and Machine Vision.* Chapman & Hall, 1993.

[16] J.W.Patterson & P.J.Willis. Computer assisted animatioon: 2d or not 2d? In N. Warren, editor, *Advances In Cross Cultural Psychology.* Halsted Press, New York, United States of America, 1975.

[17] P.Schneider. An algorithmfor automatically fitting digitized curves. In A.S.Glassner, editor, *Graphic Gems.* Academic Press, San Diego, USA, 1990.

[18] R.C.Gonzalez and P.Wintz. *Digital Image Processing.* Addison Wesley, 1987.

[19] T.Wakayama. A core-line tracing algorithm based on maximal square moving. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 68–74, January 1982.

[20] T.White. *The Animator's Workbook.* Watson Guptill, 1986.

[21] T.W.Sederberg. A physically based approach to 2-d shape blending. In *Siggrapgh Computer Graphics*, pages 25–34, July 1992.

[22] G.Wang T.W.Sederberg, P.Gao and H.Mu. 2-d shape blending: An intrinsic solution to the vertex path problem. In *Siggrapgh Computer Graphics*, pages 15–18, August 1993.

[23] V.Ranjan and A.Fournier. Matching and interpolation of shapes using union of circles. In *Computer Graphics Forum*, pages C–129–C–142, August 1996.

[24] W.T.Reeves. Inbetweening for computer animation utilizing moving point constraints. In *Siggrapgh Computer Graphics*, pages 263–269, August 1981.

[25] Jinhui Yu. *Inbetweening for computer Animation using Polar Coordinate Linear Interpolation.* University of Glasgow, Department of Computing Science, 1990.