Scientific Visualisation 3 1999
Assignment 2

Kevin Pulo
kev@zip.com.au
Student ID: 9719113

October 14, 1999

1 Vector Visualisation

1.1 Appropriate visualisation techniques

1.1.1 Hedgehog

The use of *hedgehogs* involves placing arrows in the dataset at various points, where the length and direction of the arrow directly corresponds to the length and direction of the vector at that point (interpolating if neccessary). Further, the arrows can be coloured, according to some other variable, most commonly the magnitude of the vector at that point.

Hedgehogs have the advantage that they are very easy to construct, both in terms of human effort and computational complexity. It is very easy to construct a simple network with which to visualise data using hedgehogs and the visualisation should proceed quite quickly. This allows good interaction and probing of the data and is excellent for obtaining a first look at vector data. Further, arrows are the canonical way of representing vector data, which makes them comfortable to work with.

There are disadvantages, however. The hedgehogs can be quite visually complex, which means that usually only small subsection of the data may be hedgehogged at any one time (for example, a single slice as opposed an entire data cube). Further, the use of colour to represent magnitude, while very effective, can be an additional redundancy. Hedgehogs are also quite poor at showing the vorticity (twist) of a vector field.

1.1.2 Streamlines

Streamlines attempt to address some of the problems of hedgehog. Streamlines are the lines that massless particles would follow if introduced into the vector field. Again, they are usually coloured according to the vector magnitude.

Streamlines are also easy to construct, but are more computationally expensive than hedgehogs. They provide an excellent, easily understood view of the direction of the vector field, and the colour provides the magnitude well without redundancy.

However, streamlines have no inherant direction when displayed visually (unlike hedgehogs), and as such can be ambiguous as to which direction the massless particles are actually flowing. Also, while streamlines are generally less visually complex as hedgehogs, they may still be too visually complex to allow the viewing of more than a few slices of a data cube. Since streamlines are thin lines, they are worse than hedgehogs in showing vorticity. The extension to streamlines called *ribbons* creates wide streamlines, which can twist to show the vorticity of the field.

1.1.3 Animated streamlines and particle advection

Animated streamlines are an extension to standard streamlines which are very similar to *particle advection*. Particle advection is an extension to streamlines where massless particles are animated travelling along the length of the streamlines. Animated streamlines are similar, except that the streamlines themselves are animated, with a small segment of the lines being displayed travelling along the line, from start to finish. As usual, the streamlines and particles or streamline segments may be coloured according to the magnitude of the vector field at that point.

This provides a powerful dynamic aid to streamlines. It resolves the ambiguity of the direction of the particles and also shows the strength of the field with the velocity of the particles.

Further, if the particles or streamline segments are displayed on an appropriately small scale, then whole data volumes may be visualised, although depth aids may be required. Animated streamlines and particle advection provide for observing an entire data cube of moving particles or line segments as they move, which neither hedgehogs nor static streamlines can provide. When combined with the interactivity of being able to move the viewpoint, they are nearly unparalleled in terms of observing how particles in the system would move in the vector field.

However, when visualising an entire data cube care must be taken to not overload the user with information and visual complexity. *Depth cueing* is a simple shading technique which essentially adds a “fog” to the scene, allowing the user to easily see how far away objects are. Since the data content is quite high, and these techniques are based on streamlining, they can be quite computationally expensive, which is a factor to consider when attempting interactivity.

1.2 AVS 5 networks for vector visualisation

1.2.1 Hedgehog

The network illustrating the use of the AVS 5 `hedgehog` module is contained in the file `~kpulo/ass2a.net`, and is shown in Figure 1.

The `hedgehog` module accepts three main inputs — the 3D vector field to visualise and a corresponding 3D scalar field and colourmap, with which to colour the vectors. The colours are generated from the magnitude of the vectors, with the colour range (a standard AVS colour ramp) being applied between the minimum and maximum values of the vector magnitudes (this is the function of the `minmax` and `color range` modules).

The `tube` module converts lines, that is, the vectors and their arrowheads, into 8 sided cylinders. This makes the visualisation slightly slower (and this impedes interactivity slighly), but gives the vectors a much more “solid” look, as would be desired when outputting to video or giving a presentation.

The `hedgehog` module also provides interactivity, where the plane of vectors to be visualised can be manipulated in the `geometry viewer` window. This allows the user to examine the data set by moving the plane around in the data cube.

1.2.2 Streamlines

The network illustrating the use of the AVS 5 `streamlines` module for a static streamline display is contained in the file `~kpulo/ass2b.net`, and is shown in Figure 2.

As can be seen, Figures 1 and 2 are quite similar. This is because the `hedgehog` and `streamlines` modules both deal with the same task — taking a 3D vector field and converting it to a (possibly coloured) geometry of some description. The colours

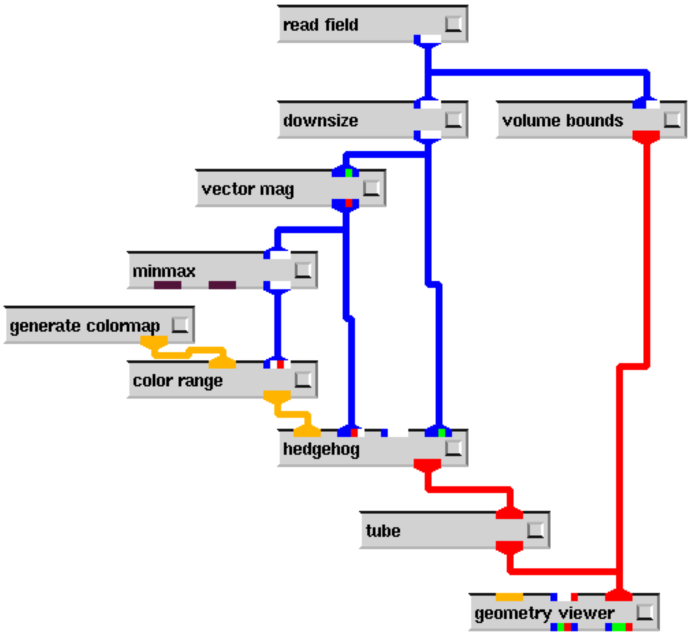


Figure 1: The AVS 5 network used to construct the hedgehog vector visualisation.

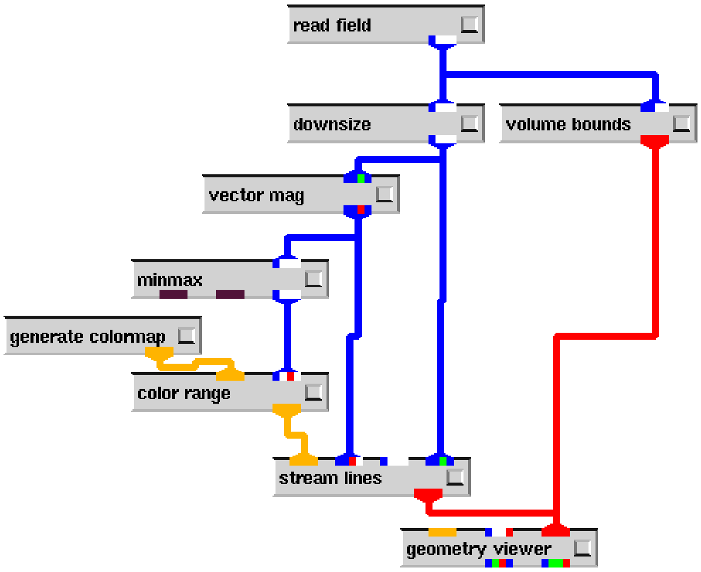


Figure 2: The AVS 5 network used to construct the streamlines vector visualisation.

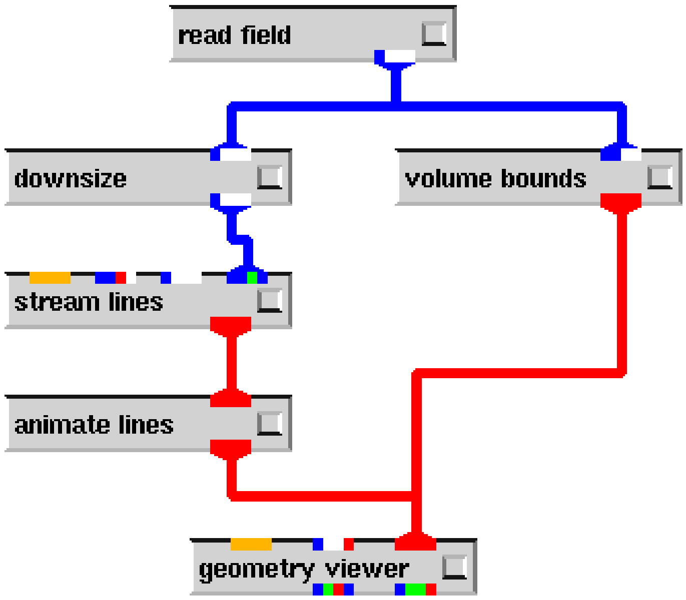


Figure 3: The AVS 5 network used to construct the animated streamlines vector visualisation.

are generated in exactly the same fashion as in Figure 1, and the `tube` module has been omitted[1].

However, the plane of sampled data points used by the `streamlines` module is different. Here we take a vertical cross-section of the data set, and can clearly see the strong jetstream at the top, the gentler winds travelling in the opposite direction at the bottom, and the turbulent region in between where wind rises into the jetstream.

1.2.3 Animated streamlines and particle advection

The network illustrating the use of the AVS 5 `streamlines` and `animate lines` module is contained in the file `~kpulo/ass2c.net`, and is shown in Figure 3.

As can be seen, the network for the animated streamlines is substantially simpler than those in Figures 1 and 2. The reason for this is that the `animate lines` module unfortunately discards any colouring information given to it by the `streamlines` module. This is a limitation of AVS 5, a better quality `animate lines` module could overcome it. As such, the colour related parts of the network were removed, with the added benefit that this improves performance.

Streamlines are generated for the entire data cube of wind velocity data. This requires a reasonably fast machine to process effectively, such as one of Vislab's O_2 machines. Performance on other, slower machines, especially where interactivity is concerned, is suboptimal.

To animate the streamlines, select the control panel for the `animate lines` module and press the `Animate` button. The streamlines will then be animated as short segments, from start to finish of each streamline. The Geometry Viewer allows the view to be rotated and manipulated while the streamlines are animating. This

[1] The `tube` module could of couse be added, though it is less useful with streamlines and again impedes the performance of the visualisation.

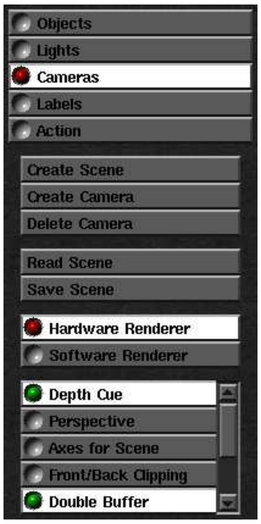


Figure 4: Enabling and disabling depth cueing in the AVS 5 Geometry Viewer.

can be exploited to great advantage to get a very good understanding of where exactly in the cube the streamlines are flowing to and from.

Alternatively, the Geometry Viewer's *Store Frames* feature can be used to store the set of frames generated when the `Animate` button is pressed, and then replayed at the full speed of the graphics system. However, this is often too fast, and AVS 5 offers no way to control the speed of Store Frames playback.

Again, when the streamlines are being animated, it is very easy to see the strong jetstream (though this time it is at the bottom), the gentler winds in the other direction and the turbulent region in between.

Depth cueing is also enabled by default. This allows the user to see very clearly the closest objects, and their relative distances. In this setting, depth cueing is almost essential to see where streamlines (and streamline segments) belong in the cube. Turning off depth cueing can help to give insights when animating the streamlines, but in a static context disabling depth cueing merely confuses the visualisation.

Unfortunately, AVS 5 doesn't provide any controls to alter the amount of depth cueing done. The depth cueing in AVS 5 seems to drop off too quickly, completely obscuring the objects at the back. What would be ideal is a control to change how far back one can see before objects are completely obscured. Figure 4 shows the AVS control panel for enabling and disabling depth cueing in the Geometry Viewer.

An alternative to depth cueing is a direct stereo view, on machines which support it[2]. In this mode, the system presents a binocularly disparite view of the geometry, typically using CrystalEyes shutter glasses. This gives the illusion of depth, and hence is quite good when visualising a cube of data such as this.

To enable full-screen stereo mode in AVS 5, hold the Control key whilst left mouse clicking in the Geometry Viewer. To disable stereo mode, press Control-left-click again. If AVS 5 does not properly restore the video mode, type `m1280` at a Unix prompt[3]. Note also that depth cueing does not work with stereo mode, and must be disabled as detailed above before entering stereo mode.

[2]At Vislab Physics, this is lewis and wilson, and at Vislab ATP, this is bruce, rrap and lee.

[3]This doesn't work on lewis. Instead, use `setmon -n 60hz`

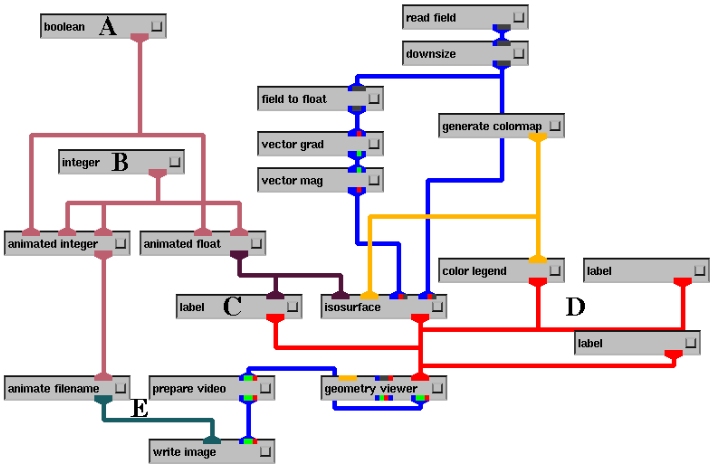


Figure 5: The AVS 5 network used to create the short lobster animation.

2 Computer Animation

2.1 AVS 5 network for an animation

Figure 5 shows the AVS 5 network used to generate the computer animation (stored in `~kpulo/ass2d.net`), and is based on `~kpulo/ass1b.net` from my SV3 Assignment 1.

This animation shows an animated isosurface of a CT/MRI scan of a lobster. It shows what is essentially a "skin" of the lobster, where the animation successively removes layers of constant density. The isosurface is coloured according to the rate of change of density, with the darker red colours indicating an approximately constant density, and lighter yellows indicating faster changing density[4]. The animation consists of two parts, the first an initial overview of the density range of the lobster, and the second a slower motion of the main transition period.

The extensions to the network which produce the animation are the modules labelled A to E in Figure 5.

The `boolean` module labelled A is the trigger for the `animated integer` and `animated float` modules, that is, switching A from on to off will commence the animation. The `integer` module labelled B is the number of steps the animation should take, and again, this sets parameters in the `animated integer` and the `animated float` modules. The `animated integer` provides the numbering for the output image files, whereas the `animated float` provides the input for the value of the `isosurface` module.

The `label` module labelled C, along with the `color legend` module and two `label` modules labelled D supplement the geometry output from the `isosurface` module. The `label` module labelled C provides the dynamic isosurface value in the output, whereas the modules labelled D provide the additional text and color legend in the output.

The `animate filename`, `prepare video` and `write image` modules, labelled E, actually perform the output. The `animate filename` module uses the output of the `animated integer` module to give a different filename for each frame to the `write image` module. The `prepare video` module takes the image from the

[4]It is easy for one's eyes and brain to see the images produced and associate the red regions with (exo-)skeletal regions, and the yellow regions with soft tissue. This is not the case; one must be careful to prevent this misinterpretation.

`geometry viewer`, optimses the image for video, and then gives this optimised image to the `write image` module, where it is written to disk.

2.2 Creating the playable animation

2.2.1 Construction of the animation

As detailed in Section 2.1, a series of numbered images were generated by the network. These are in AVS's .X format, and so must first be converted to a useful format, such as SGI's .SGI (or .RGB) format. This was acheived with the use of the MICA utility, and a command such as:

```
mica -to sgi ass2d-1.*.x
```

which converted the files `ass2d-1.001.x` – `ass2d-1.020.x` to `ass2d-1.001.sgi` – `ass2d-1.020.sgi`.

The next step was to assemble these images into a playable animation. This was acheived with the use of the MAKEMOVIE utility, and a command such as:

```
makemovie -o ass2d-1.mov -f qt -c qt_anim -r 4 -s 768,576 -b ass2d-1.*.sgi
```

which created the movie file `ass2d-1.mov` from the image files. The options passed to MAKEMOVIE are now explained.

-o ass2d-1.mov specifies the output file, in this case, `ass2d-1.mov`.

-f qt specifies the output movie format, in this case, Quicktime.

-c qt_anim specifies the compression to be used in the output movie format, in this case, Quicktime animation compression (as opposed to Quicktime video compression, which is less appropriate to this kind of animation).

-r 4 specifies the number of input images to display in the output movie per second. In this case, four input frames will be displayed per second, that is, each image will be on the screen for a quarter of a second. This makes the 20 frame animation 5 seconds long, and yet still acceptably smooth.

-s 768,576 specifies the resolution of the output movie, in this case, PAL resolution, 768x576. If the input images are smaller than this resolution (as they were), they will be scaled as much as possible while maintaining their aspect ratio, and black borders will be added to the horizontal or vertical edges as needed[5].

-b specifies that the images are to be blurred by a 1-2-1 convolution in the time dimension, which helps to reduce flickering when played back through video. This option makes no discernible difference when viewing on a computer workstation.

In a similar fashion, the slow-motion section is obtained from `ass2d-2.001.x` – `ass2d-2.020.x` to the movie file `ass2d-2.mov`.

Finally, the MOVIEMAKER utility was used to combine the two movies `ass2d-1.mov` and `ass2d-2.mov` into one movie file, `ass2d.mov`, complete with titles.

[5]This technique is called "letterboxing". It is commonly found when movies recorded for wide screen cinemas are converted to video and television formats.

2.2.2 Viewing the animation

The animation is stored in `~kpulo/ass2d.mov` as a Quicktime movie. To play this animation from a Vislab workstation, simply type the following command from a Unix prompt.

```
movieplayer /home/kpulo/ass2d.mov
```

2.3 Issues involved in the creation of computer animations

Many issues must be considered when creating a computer animation, however most fall into three broad categories: Image generation, video conversion and animation presentation.

2.3.1 Image generation

Image generation deals with obtaining the individual frames of the animation. This usually involves adding extra features to an existing visualisation to make it dynamic and to save these frames to disk in some format.

Where the data being visualised has a temporal dimension, or where time is being used as a visualisation dimension, the most common and simplest way of generating images is to simply time step through the data. The time step delta used is usually the sampled rate of the data, but may be larger than this if interpolation is acceptable[6].

The main disadvantage with this method is the frame rate is fixed at the rate the images are generated at. When the number of generated images is low, this can result in an animation that isn't smooth, however generating many frames can be time-consuming (and the problem remains that the number of frames is fixed).

The more sophisticated solution to this problem is to use a method of animating called key-frame animation. This is similar to the above method, except that only certain "key" (important) frames are chosen and the system's state at these frames stored. Frames required between these key-frames are interpolated from the system's state at the key-frames, usually linearly[7]. This has the advantage that the animation can be created from the key-frames at the frame rate of the output device. However, the disadvantage is that key-frames and the interpolation method should be chosen carefully, especially in a scientific visualisation setting, to avoid inappropriate or misleading animations from being generated.

In turn, key-frame animation leads on to the still more sophisticated areas of kinematics and dynamics (and inverse kinematics and dynamics). These are methods of simulating a system according to physical laws to provide a still more accurate and visually appealing animation.

Where frames are generated from the visualisation (in any of the above techniques), the problem of temporal aliasing occasionally comes up. This is where phenomena in the visualisation are occuring on a time-scale smaller than the time step delta of the animation. The result is that incorrect or misleading animations may be generated, with the common example being the way human eyes may perceive the wheels of a travelling car to be spinning backwards when they are actually spinning forwards at the correct frequency.

[6] Note that interpolated time step deltas smaller than the sampled rate can introduce substantialy problems due to the effect of temporal aliasing — see Section 2.3.2.

[7] The method of generating these intermediate frames is often called *tweening*.

2.3.2 Video conversion

Video conversion is the process of converting a computer animation, possibly in individual image files, obtained from one of the processes in Section 2.3.1, into a format suitable for use with video systems. The video systems refers to all manner of video equipment, but mainly storage devices (such as VHS or Beta video cassettes), display devices (such as CRTs and projectors) and processing systems (such as those for adding effects, dubbing, etc).

The first step is to process any individual image files into a playable animation, such as Quicktime or MPEG. This is usually acheived with some small utilities to ensure that each image of the visualisation is held on the display for longer than a single frame.

The resolution of the video system must be considered. Animations larger than the video resolution must be scaled down to fit, and animations smaller than the video resolution must be scaled up fit. If the aspect ratio of the animation is different to the aspect ratio of the video system, then a process called *letterboxing* is used. This adds black bars to the horizontal or vertical edges of the animation as needed. A typical video resolution is 768x576, used by PAL. This has an aspect ratio of 4:3, which is fortunately the same as most computer displays.

Video systems have what is termed a *safe area*, regions outside this area cannot be guaranteed to be displayed on all video displays. When preparing videos care must be taken to not go to the very edge of the display. Refer to Section 5.5.1 for more details.

The *frame rate* (or *fps*, for frames-per-second) of the video system is important. Most video systems have a frame rate of about 25 (PAL) or 30 (NTSC) fps. As detailed above, it is impractical to generate 25 images of the visualisation for each second of animation, hence each frame (of the animation) must be held for potentially several frames of video.

Video systems are usually interlaced, in an effort to reduce movie flicker. Interlacing is where first the odd (or even) scanlines are displayed, followed by the even (or odd) scanlines. This reduces flicker, but means that thin lines on the screen, which are one or two pixels wide (especially horizontal lines) often flicker or seem to "wobble" on the screen. As such, video preparation involves making thin lines slightly thicker to avoid this problem.

Video systems generally have a smaller dynamic colour range than computer systems, and are not normalised in the same way, either. As such, the colours must be adjusted to compensate for this. This usually involves a low-pass filter to remove small changes in colours (ie. similar colours become the same) and also increases contrast. In addition, the gamma of the image is also increased to compensate for the loss in colour range.

2.3.3 Animation presentation

The animation presentation deals with the parts of the animation which more directly affect its interpretation. They consist mainly of rules of thumb which, when followed, can aid greatly in the interpretive power of the video.

It is important that the animation enhances the information and interpretation of the data it presents. An animation of only 4 or 5 frames may be better displayed as a series of still images. For these reasons, simplicity and elegance in design are important, the animation should not crowd too much information into any one screen, and unneccessary glitz should be avoided.

Ensure that the viewers of the video have time to see, read and interpret any information presented. Text should remain present for ample time, and should be large enough to read comfortably. Similarly, animations should not proceed too

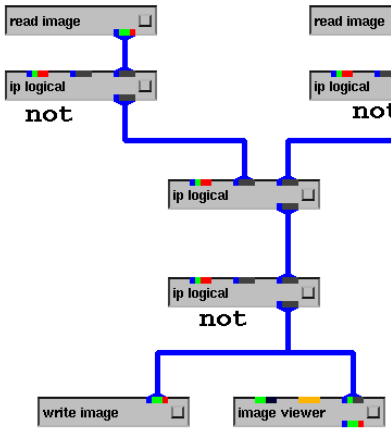


Figure 6: The AVS 5 network used to generate Figure 8 from Figure 7.

quickly.

Context is vital. Every frame of the animation should have enough context so that it could be viewed out of the animation without any extra support. In addition, the video as a whole should be self-supporting and self-documenting where possible, so that it may be viewed without any extra support. This usually involves adding titles and legends, as well as giving the viewer feedback on any currently animating values, most notably time. This feedback can be simply numbers updating, or it may be more graphical, such as a bar progressing across the screen.

A voiceover also helps in these regards, by helping the video become self-supporting, and avoiding problems that may arise when trying to narrate the video personally. Any serious video should consider using a voiceover.

Another handy feature is to show a broad overview of the entire animation at full speed, giving viewers an overview and context, and then subsequently zooming in and/or running the animation in slow-motion to concentrate on the most important parts.

3 Boolean Operator Diagrams

In all of these figures, a white pixel indicates a 0 value, and a black pixel indicates a 1 value[8]. The images were generated using the AVS network shown in Figure 6, using the ip_logical image processing module. However, this module considers white pixels to be 1 and black pixels to be 0; this is the reason for the ip_logical NOT modules just after reading the input images and just before writing the output images.

Figure 7 shows the input images, each contains a single circle at a different offset. The region inside the circle is defined to be 1, the region outside is 0.

Figure 8 illustrates the effect of combining Figures 7a and 7b using the boolean operators AND, NAND, OR, NOR, and XOR respectively.

Feature to note are:

- Figure 8a, depicting AND, is black in the region where both of the input images in Figure 7 are black. This is consistent with the definition of the AND operation.

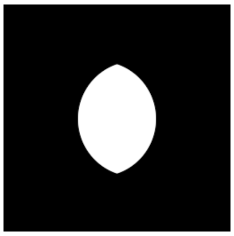
[8]This was chosen as the intended output medium is paper, where black as 1 is much more natural. As detailed, AVS works in the opposite manner, and so the images had to be converted (inversed, the application of a logical not operationo) for use with AVS.

(a) left

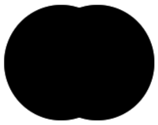
(b) right

Figure 7: The boolean operator input images.

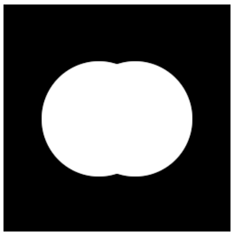
(a) and



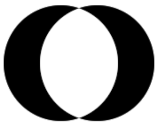
(b) nand



(c) or



(d) nor



(e) xor

Figure 8: The boolean operator output images, the result of combining Figures 7a and 7b with the indicated logical operators.

- Figure 8c, depicting OR, is black in the region where either of the input images in Figure 7 are black. This is consistent with the definition of the OR operation.
- Figure 8e, depicting XOR, is black in the regions where only one, but not both, of the input images in Figure 7 are black. This is consistent with the definition of the XOR operation.
- NAND is, by definition, a negated AND, and, similarly, NOR is a negated OR. This is reflected in Figure 8 by these two pairs or images (Figures 8a – 8b, and 8c – 8d) being "opposite", in the sense that what is black in one is white in the other, and vice-verso.

These diagrams are also called VENN DIAGRAMS (John Venn, 1834–1923, [1, Section 2.3, Page 13]), and are frequently used as demonstrative aids in elementary boolean logic and set theory (where AND corresponds to set intersection and OR corresponds to set union).

4 Surface Normal Vector of a Triangle

We denote vectors in 3-Space (3-vectors) by using boldface, and their x, y and z coordinates with subscripts. For example, the vector $\mathbf{v}$ has coordinates v_x, v_y and v_z.

The material in this section can be found in any elementary text on vectors, such as [2, Section 5.1, Page 161] or [3, Chapter 11].

4.1 Vector treatment

We are given three 3-vectors $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, defining the three points vertices of the triangle. Using elementary vector subtraction, we find two edges of the triangle, $\mathbf{E}_1$ and $\mathbf{E}_2$, using Equations 1 and 2.

$$\mathbf{E}_1 = \mathbf{P}_3 - \mathbf{P}_2 \tag{1}$$

$$\mathbf{E}_2 = \mathbf{P}_2 - \mathbf{P}_1 \tag{2}$$

The definition of the surface normal vector is a unit-length vector which is perpendicular to any vector in the plane of the surface. Thus an unnormalised surface normal vector can be obtained by taking the cross-product of any two vectors in the plane of the surface. As $\mathbf{E}_1$ and $\mathbf{E}_2$ are both vectors in the plane of the surface, Equation 3 gives one such unnormalised surface normal vector.

$$\mathbf{n} = \mathbf{E}_1 \times \mathbf{E}_2 \tag{3}$$

The unnormalised surface normal vector of Equation 3 is now normalised in Equation 4, where $|\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$ is used to denote the magnitude of the vector $\mathbf{v}$.

$$\hat{\mathbf{n}} = \frac{1}{|\mathbf{n}|}\mathbf{n} \tag{4}$$

If the points $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ are defined in a clockwise order and the surface normal is required to be in the "upward" direction, as depicted in Figure 9, then the vectors $\mathbf{E}_1$, $\mathbf{E}_2$ and $\hat{\mathbf{n}}$ form a right-handed set, as is usually desired.

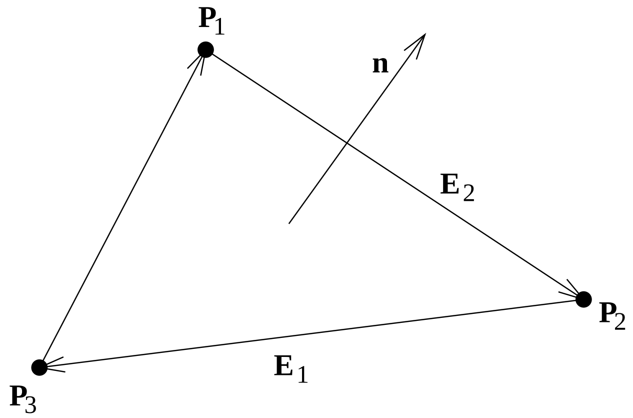


Figure 9: Geometry of the given triangle and its surface normal.

4.2 Scalar treatment

Combining Equations 1, 2 and 3, we obtain Equation 5.

$$\mathbf{n} = (\mathbf{P}_3 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_1) \tag{5}$$

Expanding this into the individual components, and using the definition of vector cross-product, we obtain the three Equations 6 to 8.

$$\begin{aligned} n_x &= (P_{3y} - P_{2y})(P_{2z} - P_{1z}) - (P_{3z} - P_{2z})(P_{2y} - P_{1y}) && (6)\\ n_y &= (P_{3z} - P_{2z})(P_{2x} - P_{1x}) - (P_{3x} - P_{2x})(P_{2z} - P_{1z}) && (7)\\ n_z &= (P_{3x} - P_{2x})(P_{2y} - P_{1y}) - (P_{3y} - P_{2y})(P_{2x} - P_{1x}) && (8) \end{aligned}$$

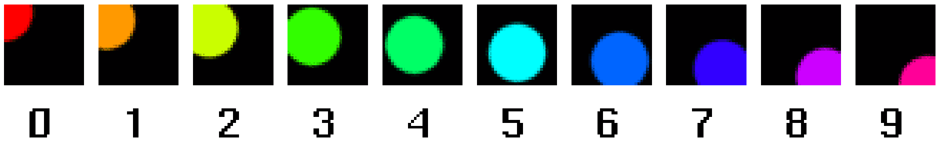
In order to normalise these values, we find the scalar value n, using the values for n_x, n_y and n_z found in Equations 6 to 8.

$$n = |\mathbf{n}| = \sqrt{n_x^2 + n_y^2 + n_z^2} \tag{9}$$

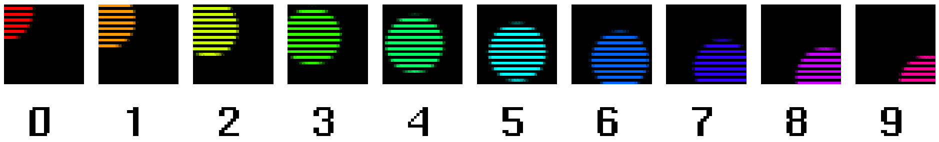
We now use this normalisation factor to find the normalised versions of Equations 6 to 8.

$$\begin{aligned} \hat{n}_x &= \frac{1}{n} n_x \\ &= \frac{1}{n}(P_{3y} - P_{2y})(P_{2z} - P_{1z}) - \frac{1}{n}(P_{3z} - P_{2z})(P_{2y} - P_{1y}) && (10)\\ \hat{n}_y &= \frac{1}{n} n_y \\ &= \frac{1}{n}(P_{3z} - P_{2z})(P_{2x} - P_{1x}) - \frac{1}{n}(P_{3x} - P_{2x})(P_{2z} - P_{1z}) && (11)\\ \hat{n}_z &= \frac{1}{n} n_z \\ &= \frac{1}{n}(P_{3x} - P_{2x})(P_{2y} - P_{1y}) - \frac{1}{n}(P_{3y} - P_{2y})(P_{2x} - P_{1x}) && (12) \end{aligned}$$

The values $\hat{n}_x$, $\hat{n}_y$ and $\hat{n}_z$ given in Equations 10 to 12 are now the x, y and z coordinates of the normalised surface normal to the triangle with vertices $\mathbf{P}_1 = (P_{1x}, P_{1y}, P_{1z})$, $\mathbf{P}_2 = (P_{2x}, P_{2y}, P_{2z})$ and $\mathbf{P}_3 = (P_{3x}, P_{3y}, P_{3z})$.



(a) Graphics frames



(b) Video fields

Figure 10: The 10 frames and fields of the moving ball, as taken by a graphics system and video system, respectively.

5 Differences Between Computer Graphics Systems and Video Systems

5.1 Fields, frames and interlacing

The main, fundamental difference between computer graphics systems and video systems is the way they store images.

Computer graphics systems store *frames*, where a frame is all the data required for a single image. However, video systems store *fields*, where a field is image data sampled at a (single) point in time; this is usually half the data in a frame. The field rate of a video signal is the rate of field sampling, and determines the temporal difference of the fields.

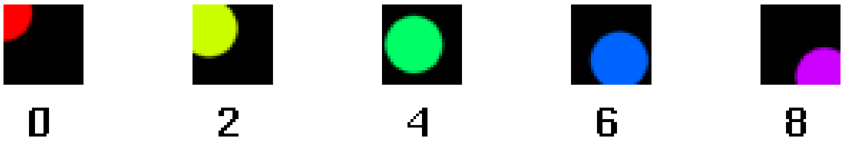
Fields are usually *even interlaced*, such that the half of the frame they store is initially the even scanlines of the frame , followed by the odd scanlines, and so on[9].

Video systems do not support any real notion of frames. The idea that a video frame is two adjacent fields is wrong, and is a result of trying to apply computer graphic frame concepts to video, where they don't apply. However, it is occasionally useful to think of a frame as two adjacent fields, ignoring the temporal difference between them.

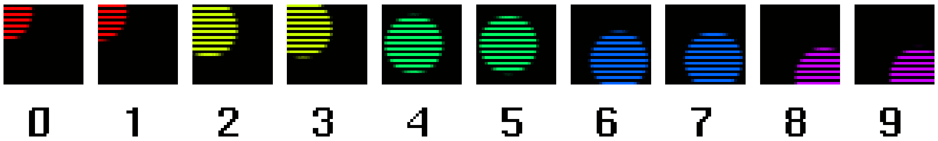
As a result, there is no way to recover the data for a full frame at a given time instant from a video signal. Simply half the visual data is missing from each time instant recorded.

Reference [4] has an excellent illustration of the difference between fields and frames. The scenario is a moving ball being recorded by a film camera (ie. as frames), shown in Figure 10(a), and by a video camera (ie. as fields), shown in Figure 10(b). Notice that in Figure 10(b) each field has a different spatial location, that is, each field is of a different instant in time. By contrast, Figure 11 shows two ways that the video camera will *not* record the moving ball. Figure 11(a) shows full frames at half the field rate, this is incorrect because video cameras record fields at

[9]Notably, the PAL system uses even interlacing, whereas NTSC uses odd interlacing, that is, odd scanlines followed by even scanlines.



(a) Frames, at half the frame rate



(b) Fields, each pair of fields making up a frame

Figure 11: Examples of how the video recording will not look like.

their field rate. Figure 11(b) shows fields where each pair of fields together make up a single frame, this is incorrect because the field pairs are of the same instant in time, whereas the video camera will capture each frame at a different instant in time, as in Figure 10(b).

5.2 Refresh rate and interlacing revisted

The refresh rate is the rate at which the display is updated. For field-based systems, this is the rate at which fields are displayed, and for frame-based systems it is the rate at which the frames are displayed. These two methods are called *interlaced scanning* and *progressive scanning*, respectively. Interlaced scanning updates the odd or even lines for a field once per refresh, whereas progressive scanning updates all the scanlines once per refresh. Hence video systems use interlaced scanning, whereas computer graphics systems use progressive scanning[10].

5.2.1 Frame rate

Discussion of the frame rate of a system has been delayed until this point, where frames, fields, field rate, refresh rate and interlacing have been discussed.

The frame rate of a graphics or video system is the rate at which frames are displayed. Since a frame is a complete image, another way of stating this is the frame rate is the rate at which scanlines are updated. For a graphics system using progressive scanning, this is simply its refresh rate. However, for a video system (which uses interlaced scanning), this is half the field rate, since a field contains half the scanlines of the display. For example, NTSC systems have a field rate (and hence refresh rate) of 60Hz[11], so they have a frame rate of 30Hz.

[10] Computer graphics systems do sometimes offer interlaced scanning when displaying, however, this is still fundamentally a frame-based system. With interlaced computer graphics systems, the odd and even scanlines are displayed for a single frame, that is, a single instant in time. However, with video systems, the interlaced scanning displays the odd and even scanlines of different fields, and hence different instants in time.

[11] Actually, 59.94Hz.

Since human persistance of vision is approximately 1/25th second, frame rates should ideally be 25Hz or greater to avoid noticeable flicker[12]. All the major video formats, that is, PAL, NTSC and HDTV have frame rates greater than or equal to 25Hz.

5.3 Colour

The human colour perception system uses three varieties of cones in the retina, each responding to a different wavelength spectrum. This is known as tristimulus colour, and is the reason colour specifications require three attributes to accurately describe an arbitrary colour.

There are two main types of colour systems employed in video and graphic systems, *component colour systems* and *composite colour systems*.

Component systems transmit independant colour signals, and such systems include the additive RGB system and the subtractive CMY(K) system. Component colour systems are most often found in graphics systems.

Composite systems use frequency interleaving to combine three attributes into a single signal for transmission. The three attributes are a luminance and two colour differences. The luminance is a measure of the brightness of the signal (and so alone is suitable for black and white displays), and the colour differences are the difference between two primary colour values, such as the red minus the yellow components. The two most common examples are NTSC's YIQ (Luminance (Y), Interphase (I) and Quadrature (Q)) and PAL's YUV. Composite systems are most often found in video systems as they use less bandwidth (see Section 5.5.2) than component systems with no loss of information. In addition, the colour difference signals can be modulated onto the primary luminance signal, allowing backward-compatible introduction of colour television sets.

The colour gamuts of the composite video systems are also such that high saturation[13] colours are to be avoided. NTSC has problems with colours of high saturation and low luminance, and complementary saturated colours may colour-bleed on screen and videotape. Also, the colour gamut of PAL is smaller than that of NTSC, so some colours visible in NTSC cannot accurately be reproduced in PAL.

5.4 Resolution and aspect ratio

The resolution of a given system is the dimensions in pixels of the displayed image, and the aspect ratio is the ratio of width to height. The resolution is a major factor in image quality, and aspect ratio must be preserved when transferring images to prevent them from becoming distorted.

Most video and graphics systems share the same aspect ratio, 4:3. This removes some of the complexity when transferring images, and even when aspect ratios differ the technique of letterboxing may be used to add black borders when neccessary to preserve the aspect ratio of the original image.

PAL has a 625 line frame, however, 8% of this is used for synchronisation[14], and this gives an effective resolution of 768 x 576. NTSC has a 525 line frame, but again 8% is used for synchronisation, giving an effective resolution of 640 x 480. Typical computer graphics frame resolutions are 640 x 480, 800 x 600, 1024 x 768, 1152 x 864, 1280 x 1024 and 1600 x 1200.

[12]Note, however, that flicker can have many other causes, such as properties of the monitor's phosphors and properties of the video camera.

[13]Saturation is the colourfulness of an area relative to its brightness.

[14]And other information, for example, closed-captioning.

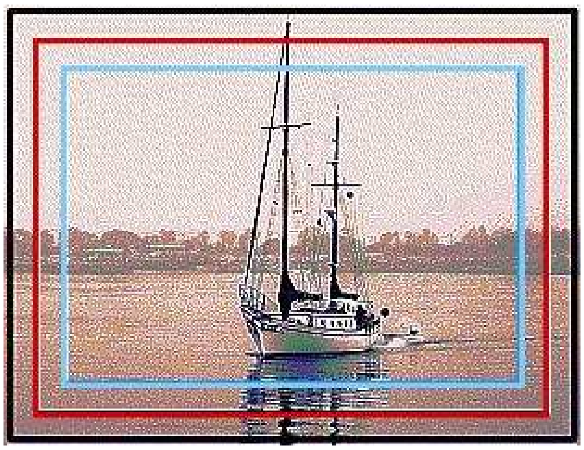


Figure 12: The safe areas of a video display, on a sample image. The inner line indicates the border of the safe title area, the next most inner line indicates the border of the safe action area and the outer line indicates the extent of the video frame.

5.5 Miscellaneous issues

5.5.1 Safe areas

Most video monitors suffer from a problem called *overscanning*, which is where the image is actually displayed past the edges of the visible section of the screen[15]. As a result the parts of the video image very close to the edge may be lost to this overscanning.

To combat this problem there exists two *safe areas* of video. The first is the *safe action area* (also known as the *essential area*), this area can be displayed by the vast majority of video monitors and is suitable for motion and action displays. The second is the *safe title area*, which is the area which can safely be used for undistorted, unobstructed text and similar displays.

As a general rule of thumb, the central 80% of the display is the safe title area and the central 90% of the display is the safe action area. Figure 12 illustrates these areas on a sample image.

5.5.2 Bandwidth

The *bandwidth* of a signal is the range of broadcast frequencies used in the transmission of the signal[16]. There is a limit to the amount of information that can be broadcast in a given bandwidth, and so the neccessary bandwidth is the minimum amount of spectrum required to transmit a given signal without distortion, degradation of other loss of information. Since frequencies in the radio spectrum are usually at a premium, techniques such as composite colour systems (as described in Section 5.3) are often desired to lower the neccessary bandwidth for a transmitted signal.

[15]Overscanning is partly a historical legacy, from when CRT displays couldn't be manufactured in rectangular shape and so were round with the round edges masked off and deliberated overscanning in these masked off areas.

[16]For example, television typically uses a bandwidth of 6 MHz per channel.

References

[1] Choo, K-G and Taylor, D. E., *Introduction to Discrete Mathematics*, Addison Wesley Longman, Melbourne, Australia, 1994.

[2] Foley, Van Dam, *et al*, *Introduction to Computer Graphics*, Addison Wesley, Reading, Massachusetts, 1994.

[3] Stewart, *Calculus, 3rd Edition*, Brooks/Cole, Pacific Grove, California, 1995.

[4] Pirazzi, Chris, *The Lurker's Guide to Video*, http://reality.sgi.com/cpirazzi_engr/lg/

[5] Network Computing Services, Inc, *Video Guide*, http://www.msc.edu/msc/docs/video/