conformational change between one node and the next. We shall call such sites and their associated matrices 'conformons'. The technique then depends on the stacking technique in which matrices are stored and later recovered in the reverse order of their storage.

One begins at some reference point deemed to be fixed in data space and at this point one stacks the prevailing viewing transformation. From this reference point one advances through the molecule along the structural tree and as each conformon is encountered its matrix is calculated. The product of the prevailing matrix with the conformon matrix is formed and stacked, and this product becomes the prevailing matrix. This product is constructed with the conformon matrix as a factor on the right, *i.e.* in data space as defined in Section 3.3.1.3.1, and is calculated using the coordinates of the molecule in their unmodified form, *i.e.* before any shape changes are brought about.

This progression leads eventually to an extremity of the tree. At this point drawing is commenced using the prevailing matrix and working backwards towards the fixed root, unstacking (or 'popping') a matrix as each conformon is passed until a node is reached, which, in general, will occur only part way back to the root. On reaching such a node drawing is suspended and one advances along the newly found branch as before, stacking matrices, until another extremity is reached when drawing towards the root is resumed. This alternation of stacking matrices while moving away from the root and drawing and unstacking matrices while moving towards the root is continued until the whole tree is traversed.

This process is illustrated schematically in Fig. 3.3.1.2 for a simple tree with one node, numbered 1, and three conformons at *a*, *b* and *c*. One enters the tree with a current viewing transformation $\boldsymbol{T}$ and progresses upwards from the fixed lower extremity. When the conformon at *a* is encountered, $\boldsymbol{T}$ is stacked and the product $\boldsymbol{TM}_a$ is formed. Continuing up the tree, at node 1 either branch may be chosen; we choose the left and, on reaching *b*, $\boldsymbol{TM}_a$ is stacked and $\boldsymbol{TM}_a\boldsymbol{M}_b$ is formed. On reaching the tip drawing down to *b* is done with this transformation, $\boldsymbol{TM}_a$ is then unstacked and drawing continues with this matrix until node 1 is reached. The other branch is then followed to *c* whereupon $\boldsymbol{TM}_a$ is again stacked and the product $\boldsymbol{TM}_a\boldsymbol{M}_c$ is formed. From the tip down as far as *c* is drawn with this matrix, whereupon $\boldsymbol{TM}_a$ is unstacked and drawing continues down to *a*, where $\boldsymbol{T}$ is unstacked before drawing the section nearest the root.

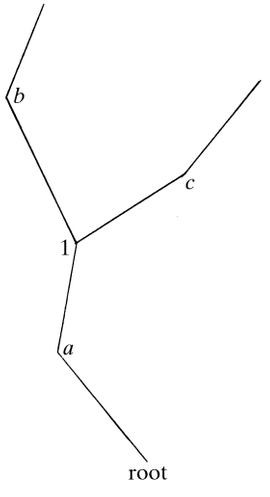

Fig. 3.3.1.2. Schematic representation of a simple branched-chain molecule with a stationary root and two extremities. The positions marked *a*, *b* and *c* are the loci of possible conformational change, here called conformons, and there is a single, numbered branch point.

With this organization the matrices associated with *b* and *c* are unaffected by changes in the conformation at *a*, notwithstanding the fact that changes at *a* alter the direction of the axis of rotation at *b* or *c*.

Two other approaches are also possible. One of these is to start at the tip of the left branch, replace the coordinates of atoms between *b* and the tip by $\boldsymbol{M}_b\mathbf{X}$, and later replace all coordinates between the tip and *a* by $\boldsymbol{M}_a\mathbf{X}$, with a similar treatment for the other branch. The advantage of this is that no storage is required for stacked matrices, but the disadvantage is that atoms near the tips of the tree have to be reprocessed for every conformon. It also modifies the stored coordinates, which may or may not be desirable.

The second alternative is to draw upwards from the root using $\boldsymbol{T}$ until *a* is reached, then using $\boldsymbol{TM}_a$ until *b* is reached, then using $\boldsymbol{TM}'_b\boldsymbol{M}_a$ to the tip, but in this formulation $\boldsymbol{M}'_b$ must be based on the geometry that exists at *b* after the transformation $\boldsymbol{M}_a$ has been applied to this region of the molecule, *i.e.* $\boldsymbol{M}'_b$ is characteristic of the final conformation rather than the initial one.

3.3.1.5. *Drawing techniques*

3.3.1.5.1. *Types of hardware*

There are two main types of graphical hardware in use for interactive work, in addition to plotters used for batch work. These main types are raster and vector. In raster equipment the screen is scanned as in television, with a grid of points, called pixels, addressed sequentially as the scan proceeds. Associated with each pixel is a word of memory, usually containing something in the range of 1 to 24 bits per pixel, which controls the colour and intensity to be displayed. Many computer terminals have one bit per pixel (said to be 'single-plane' systems) and these are essentially monochrome and have no grey scale. Four-plane systems are cheap and popular and commonly provide 4-bit by 4-bit look-up tables between the pixel memory and the monitor with one such table for each of the colours red, green and blue. If these tables are each loaded identically then 16 levels of monochrome grey scale are available, but if they are loaded differently 16 different colours are available simultaneously chosen from a total of 4096 possibilities. Four-plane systems are adequate for many applications where colour is used for coding, but are inadequate if colour is intended also to provide realism, where brilliance and saturation must be varied as well as hue. For these applications eight-plane systems are commonly used which permit 256 colours chosen from 16 million using three look-up tables, though the limitations of these can also be felt and full colour is only regarded as being available in 24-plane systems.

Raster-graphics devices are ideal for drawing objects represented by opaque surfaces which can be endowed with realistic reflecting properties (Max, 1984) and they have been successfully used to give effects of transparency. They are also capable of representing shadows, though these are generally difficult to calculate (see Section 3.3.1.5.5). Many devices of this type provide vectorization, area fill and anti-aliasing. Vectorization provides automatically for the loading of relevant pixels on a straight line between specified points. Area fill automatically fills any irregular pre-defined polygon on the screen with a uniform colour with the user specifying only the colour and one point within the polygon. Anti-aliasing is the term used for a technique which softens the staircase effect that may be seen on a line which runs at a small angle to a vertical or horizontal row of pixels.

The main drawback with this type of equipment is that it is slow compared to vector machines. Only relatively simple objects can be displayed with smooth rotation in real time as transformed coordinates have to be converted to pixel addresses and the

previous frame needs to be deleted with each new frame unless it is known that each new frame will specify every pixel. However, the technology is advancing rapidly and these restrictions are already disappearing.

Vector machines, on the other hand, are specialized to drawing straight lines between specified points by driving the electron beam along such lines. No time is wasted on blank areas of the screen. Dots may be drawn with arbitrary coordinates, in any order, but areas, if they are to be filled, must be done with a ruling technique which is very seldom done. Images produced by vector machines are naturally transparent in that foreground does not obscure background, which makes them ideal for seeing into representations of molecular structure.

3.3.1.5.2. *Optimization of line drawings*

A line drawing consisting of n line segments may be specified by anything from $(n+1)$ to $2n$ position vectors depending on whether the lines are end-to-end connected or independent. Appreciable gains in both processing time and storage requirements may be made in complicated drawings by arranging for line segments to be end-to-end connected as far as possible, and an algorithm for doing this is outlined below. For further details see Diamond (1984*a*).

Supposing that a list of nodal points (atoms if a covalent skeleton is being drawn) exists within a computer with each node appearing only once and that the line segments to be drawn between them are already determined, then at each node there are, generally, both forward and backward connections, forward connections being those to nodes further down the list. A quantity D is calculated at each node which is the number of forward connections minus the number of backward connections. At the commencement of drawing, the first connected node in the list must have a positive D, the last must have a negative D, the sum of all D values must be zero and the sum of the positive ones is the number of strokes required to draw the drawing, a 'stroke' being a sequence of end-to-end connected line segments drawn without interruption. The total number of position vectors required to specify the drawing is then the number of nodes plus the number of strokes plus the number of rings minus one.

Drawing should then be done by scanning the list of nodes from the top looking for a positive D (usually found at the first node), commencing a stroke at this node and decrementing its D value by 1. This stroke is continued from node to node using the specified connections until a negative D is encountered, at which point the stroke is terminated and the D value at the terminating node is incremented by 1. This is done even though this terminating node may also possess some forward connections, as the total number of strokes required is not minimized by keeping a stroke going as far as possible, but by terminating a stroke as soon as it reaches a node at which *some* stroke is bound to terminate.

The next stroke is initiated by resuming the scan for positive D values at the point in the node list where the previous stroke began. If this scan encounters a zero D value at a node which has not hitherto been drawn to, or drawn from, then the node concerned is isolated and not connected to any other, and such nodes may require to be drawn with some special symbol. The expression already given for the number of vectors required is valid in the presence of isolated nodes if drawing an isolated node is allowed one position vector, this vector not being counted as a stroke.

The number of strokes generated by this algorithm is sensitive to the order in which the nodes are listed, but if this resembles a natural order then the number of strokes generated is usually close to the minimum, which is half the number of nodes having an odd number of connections. For example, the letter E has six nodes, four of which have an odd number of connections, so it may be drawn with two strokes.

3.3.1.5.3. *Representation of surfaces by lines*

The commonest means of representing surfaces, especially contour surfaces, is to consider evenly spaced serial sections and to perform two-dimensional contouring on each section. Repeating this on serial sections in two other orientations then provides a good representation of the surface in three dimensions when all such contours are displayed. The density is normally cited on a grid with submultiples of **a**, **b** and **c** as grid vectors, inverse linear interpolation being used between adjacent grid points to locate points on the contour. For vector-graphics applications it is expedient to connect such points with straight lines; some equipment may be capable of connecting them with splines though this is burdensome or impossible if real-time rotation of the scene is required. Precalculation of splines stored as short vectors is always possible if the proliferation of vectors is acceptable. For efficient drawing it is necessary for the line segments of a contour to be end-to-end connected, which means that it is necessary to contour by following contours wherever they go and not by scanning the grid. Algorithms which function in this way have been given by Heap & Pink (1969) and Diamond (1982*a*). Contouring by grid scanning followed by line connection by the methods of the previous section would be possible but less efficient. Further contouring methods are described by Sutcliffe (1980) and Cockrell (1983).

For raster-graphics devices there is little disadvantage in using curved contours though many raster devices now have vectorizing hardware for loading a line of pixels given only the end points. For these devices well shaped contours may be computed readily, using only linear arithmetic and a grid-scanning approach (Gossling, 1967). Others have colour-coded each pixel according to the density, which provides a contoured visual impression without performing contouring (Hubbard, 1983).

3.3.1.5.4. *Representation of surfaces by dots*

Connolly (Langridge *et al*., 1981; Connolly, 1983*a*,*b*) represents surfaces by placing dots on the surface with an approximately uniform superficial density. Connolly's algorithm was developed to display solvent-accessible surfaces of macromolecules and provides for curved concave portions where surface atoms meet. Pearl & Honegger (1983) have developed a similar algorithm, based on a grid, which generates only convex portions which meet in cusps, but is faster to compute than the Connolly surface. Bash *et al.* (1983) have produced a van der Waals surface algorithm fast enough to permit real-time changes to the structure without tearing the surface.

It has become customary to use a dot representation to display computed surfaces, such as the surface at a van der Waals radius from atomic centres, and to use lines to represent experimentally determined surfaces, especially density contours.

3.3.1.5.5. *Representation of surfaces by shading*

Many techniques have been developed, mainly for raster-graphics devices, for representing molecular surfaces and these have been very well reviewed by Max (1984).

The simplest technique in this class consists in representing each atom by a uniform disc, or high polygon, which can be colour-coded and area-filled by the firmware of the device. If such atoms are sorted on their z coordinate and drawn in order, furthest ones first, so that nearer ones partly or completely overwrite the further ones then the result is a simple representation of the molecule as seen from the front. This technique is fast and has its uses when a rapid schematic is all that is required. In one sense it is wasteful to process distant atoms when they are going to be overwritten by foreground atoms, but front-to-back processing requires the boundaries of visible parts

references