Multidimensional Data Format Specification: A Generalization of the American College of Radiology–National Electric Manufacturers Association Standards

Jayaram K. Udupa, Hsiu-Mei Hung, Dewey Odhner, and Roberto Goncalves

Multidimensional image data are becoming increasingly common in biomedical imaging. Three-dimensional visualization and analysis techniques based on three-dimensional image data have become an established discipline in biomedicine. Some imaging problems generate image data of even higher dimensions. It often becomes necessary, rather than just convenient, to consider the higher-dimensional data as a whole to adequately answer the underlying imaging questions. Despite this established need for convenient exchange of image and image-derived information, no exchange protocols are available that adequately meet the needs of multidimensional imaging systems. This paper describes an exchange protocol that has been designed after careful consideration of the common requirements of methodologies for visualization and analysis of multidimensional data. It is based on and is a generalization of the widely accepted American College of Radiology–National Electrical Manufacturers Association (ACR-NEMA) standards specified for two-dimensional images. It is implemented and actively being used in a data-, application-, and machine-independent software environment, being developed in the authors' department, for the visualization and analysis of multidimensional images.
Copyright © 1992 by W.B. Saunders Company

MULTIDIMENSIONAL DATA arise in a variety of disciplines in the biomedical sciences. The data are either produced by an imaging device, such as an imaging scanner, or generated by a simulation procedure. Although three-dimensional (3-D) image data [generated by medical computed tomography (CT), magnetic resonance (MR), positron-emission tomography (PET), and ultrasonography (US) scanners] are the most frequently encountered, higher dimensional data are also becoming available. Some examples of n-dimensional (nD) data with $n > 3$ that are now available or that can be potentially generated are as follows. CT or MR volume images of a moving joint or of a beating heart[1,2]: $n = 4$, the independent variables being three spatial and one temporal. MR spectroscopic images[3] of a beating heart: $n = 5$, four independent variables are as in the previous example and the fifth is an MR spectral variable. Dynamic phase-space representation of electroencephalogram signals: $n = 4$ to 8,[4] an example of data generated by simulation procedures.

The usefulness of treating 3-D image data holistically rather than as a set of 2-D images has been well established by the developments in 3-D imaging during the past 15 years.[5] The need for treating higher dimensional data as a whole (rather than as a set of 3-D data) requires some justification because they are not yet as common and established as 3-D data. Of the many examples that illustrate the convenience (such as saving user time, computational time, etc) resulting from such treatment, we give two examples that illustrate the necessity of such treatment. (1) When an nD data set contains multiple structures, it is crucial to consider each structure in its entirety. Consider a 4-D data set pertaining to a beating heart. The data may contain multiple 3-D dynamic structures such as blood pools in chambers and vessels, myocardium, and so on. The connectedness of each dynamic structure is hard to infer from the substructures presented at each instant. This inference may be an impossible mental exercise when the structure does not conform to known normal configurations, as in congenital heart defects. Considering the structure as a 4-D (connected) entity may be the only approach for the success of visualization and analysis of such structures. (2) There are MR imaging protocols that allow tagging the myocardium magnetically during the imaging process.[6] This essentially allows affixing a grid system to the myocardium at some initial instant, which then moves with the myocardium during the cardiac cycle. By studying how the grid system distorts, it is possible to perform a comprehensive motion

From the Department of Radiology, University of Pennsylvania, Philadelphia, PA.

Supported by National Institutes of Health Grant CA50851.

Address reprint requests to Jayaram K. Udupa, Department of Radiology, University of Pennsylvania, Philadelphia, PA.

Copyright © 1992 by W.B. Saunders Company
0897-1889/92/0501-0009$03.00/0

analysis of the myocardium.[7] Unless some simplifying assumptions about the nature of the motion itself are made, this cannot be treated as a 2-D or 3-D image analysis problem.

It is clear that there is a need for a multidimensional data exchange format that fulfills the needs of multidimensional data visualization and analysis. For reasons given earlier, it is just as important to treat multidimensional structures derived from image data holistically as it is to consider image data in that fashion. As multidimensional data and their analysis become more common, there will be a need for exchange of structure data in addition to the need for exchange of image data. No comprehensive protocols exist yet that cater to the needs of multidimensional image visualization and analysis. This article is an attempt to fill this gap.

Although several file exchange protocols exist in the general area of imaging science,[8-10] and efforts are underway to address the data exchange issue in the more general area of scientific visualization,[11] we are not aware of any established exchange protocol that has taken into careful consideration the requirements of biomedical visualization and analysis. We have been developing a data-, application-, and machine-independent software environment, called 3DVIEWNIX,[12] for the visualization and analysis of multidimensional biomedical data. This environment incorporates a variety of methodologies and algorithms and is developed for UNIX machines[13] supporting the X-Window system.[14] Because American College of Radiology–National Electrical Manufacturers Association (ACR-NEMA) standards[15] suggested for the representation, archival, and communication of 2-D images are the most widely accepted protocols within medical imaging, and because their extension and generalization to include multidimensional image and structure data did not pose any intrinsic difficulties, we decided to build on their principles. This paper describes the principles underlying this generalization and the complete generalized specification. We have tried to make the paper self-contained and independent of the description of the ACR-NEMA specifications. Readers interested in more details of the ACR-NEMA specification may consult their publication.[15]

TERMINOLOGY

Data handled by multidimensional imaging systems are mainly of three kinds. The first can be considered to be the digitized and quantized estimate of a vector-valued function of a vector variable. The second and the third represent information derived from the first.

We call the first a *scene,* the discretized domain of the function the *scene domain,* the dimension of the independent vector variable the *scene dimension* (*sd*), the device that produces an estimate of the function an *imaging device,* each digitized instance of the independent vector variable a *cell,* and the quantized value of the function at the cell the *cell density.* As an example, a stack of slice image data captured by an imaging device such as a CT scanner is a 3-D scene where the function is scalar valued. At one end, a scene may thus represent a time sequence of MR image data pertaining to a dynamic organ such as the human heart, and at the other end it may represent simply a scalar-valued function of a scalar variable. With each scene, we associate a fixed coordinate system $x_1, x_2, \ldots, x_{sd}$ called the *scene coordinate system.* We assume a fixed coordinate system called the *imaging device coordinate system* to be associated with each imaging device. We assume these systems to be *right-handed* in the sense of a generalization of a right-handed 3-D coordinate system (ie, if $u_1, u_2, \ldots, u_{sd}$ are unit vectors along the coordinate axes, we assume $u_i \times u_{i+1} = u_{i+2}$, where $1 \leq i \leq sd$ and $u_{sd+1} = u_1$ and $u_{sd+2} = u_2$). Each cell has a unique *location* in the scene specified by its real coordinates in the scene coordinate system. We note that the unit of measurement may not be the same along the different coordinate axes.

Data *derived* from scenes may be in many forms. They may represent another scene; for example, a scene resulting from applying to a given scene a variety of scene transformations such as filtering, segmentation, and interpolation. They may represent nonscene *structures;* for example, a set of cells constituting an opaque/semitransparent object or the boundary of a set of cells. They may also represent *display* data; for example, a sequence of red, green, blue color-valued 2-D images representing renditions of a sequence of views of the surface of an object.

Associated with every *structure data* set is a right-handed *structure coordinate system* y_1, y_2, . . . , y_{sd}. We assume that all structures are representable as a tree through hierarchical subdivision, the subdivisions representing natural groupings in the structure. For example, a 3-D object represented as a set of cells can be divided into subsets, each of which represents cells having the same y_3 coordinate. Each such subset may be further divided into smaller subsets representing cells with the same y_2 coordinate. Finally, each subdivision of each smaller subset contains the individual cell (ie, the description associated with each cell). This structure represents a slice-by-slice and a row-by-row representation of an arbitrary set of cells. We call the entities represented by the leaf nodes of the tree the *terminal structure elements* and the entities represented by the intermediate groupings the *non-terminal structure elements.* In the example just given, the terminal structure elements are the cells; the slice and row groupings constitute the nonterminal structure elements, and the tree has four levels including the root (Fig 1). Most of the structures encountered in multidimensional imaging systems have characteristics similar to those in this example: the tree has a few (two to four) levels; the leaf nodes all occur at the last level; the number of nonterminal structure elements is a small fraction of the total number of terminal structure elements. However, there are other structures, such as the spatial subdivision trees (eg, octrees, quadtrees) that generally do not have these properties.

A *display data* set derived from a scene is a set of 2-D images that portray information about specific aspects of the objects captured in the scene. In general, the *intensity* associated with the pixels of the images in the display is vector valued.

In summary, data processed and generated by multidimensional imaging systems are mainly of three kinds: scene data, structure data, and display data. The first constitute mainly data generated by imaging devices or some processed version thereof, the second object information extracted from scenes, and the third a composition of specific object information ready to be displayed.

In designing a data format, one invariably encounters the issue of trade-off between the generality of format and its efficiency. In the data format proposed here, we have tried to keep the generality with as little sacrifice of efficiency as possible. This we achieve by recognizing each situation where data may inherently be of a distinct nature (from the consideration of processing done on the data) and by assigning to it a distinct data type. To do this we identify further data subtypes (IMAGE0, IMAGE1, CURVE0, SURFACE0, SURFACE1, SHELL, etc) among scene, structure, and display data. A specific meaning is imparted to the information stored in a file when we know the data type it represents.

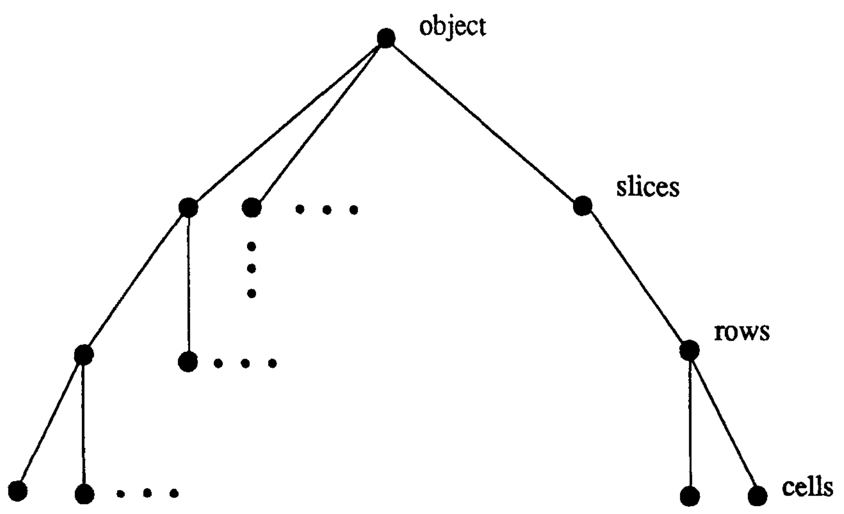

Fig 1. Structure data represented as a tree.

We use the following abbreviations, mostly as used in the ACR-NEMA standards specifications,[15] to describe the data format.

Method of value representation (VR):

BI:	16 bit binary (2's complement), the left byte being more significant than the right byte.
BD:	32 bit binary unsigned, with the bytes appearing in the order byte1, byte0, byte3, byte2, where byte0, . . . , byte3 are in the increasing order of significance. (See LIMITATIONS AND EFFICIENCY.)
GN:	Generalized numeric; a set of integer and floating point numbers. The integers are stored first followed by the floating point numbers. The integers may be packed in any specified bit fields and may be signed (2's complement) or unsigned. The floating point numbers may be single or double precision and are stored as per Institute of Electrical and Electronic Engineers standards.[16]
AN:	ASCII numeric. An integer number should contain only the characters 0 through 9, +, and −. A fixed-point number should contain only the characters 0 through 9, +, −, and an explicit decimal point (implied decimal points are not permitted). A floating-point number should be as defined in ANSI X3.9 (1978). Character E is permitted to indicate the start of an exponent. When a word consists of two characters, the first should be in the least significant byte. All ASCII fields should be padded if necessary to produce an even number of bytes, the padding character being the space (20H). The AN text should be right-justified.
AT:	ASCII text, storage as for AN except that the AT text should be left-justified.

Definition of value multiplicity (VM): S, single value; M, multiple value.

Data element types: 1, type1 (necessary for storage and display of the data); 1D, type1D (type1 with default values); 2, type2 (primary values used for interpretation); 2D, type2D (type2 with default values); 3, type3 (secondary items used for interpretation).

If VM is M when VR is AT or AN, the delimiter used to separate multiple values is "\". When VR is BI, BD, or GN, no delimiter is needed and hence not used. (In the latter case, we use delimiters in our descriptions just to show the separation of different value fields.)

DESCRIPTION OF DATA TYPES

In this section we present complete specifications of scene, structure, and display data.

Scene Data

Among scene data we identify two types—IMAGE0 and IMAGE1. For IMAGE0 type, we assume that the sampling scheme (ie, the method of digitizing the independent variable) is more systematic, in particular, slice oriented, and matches the way most imaging devices capture scene data. For IMAGE1 type, the sampling scheme is arbitrary.

IMAGE0. We assume the scene domain to be such that (1) the set S of all cells in the scene domain with fixed x_i coordinates, for $3 \leq i \leq sd$, to be a 2-D rectangular array, and (2) the distances between consecutive cells in the x_1 and x_2 directions are constants (if the units of measurement along x_1 and x_2 are different, we assume an appropriate definition of distance between cells in the (x_1, x_2) space to be given). This implies that if we consider the cells to constitute points in an sd-dimensional Euclidean space, the array defined by S constitutes a lattice of rectangles (this is equivalent to having rectangular pixels). We call S together with the densities associated with its cells a *slice.* Often data captured by imaging devices are such that the set S is not a rectangular array. Such data should be appropriately processed to bring them into this slice form. Unlike the domain of the slice, we will soon see that the scene domain itself need not be of a rectangular array form for an IMAGE0 type scene.

For a given n, $2 \leq n \leq sd$, and fixed numbers $X_{n+1}, X_{n+2}, \ldots, X_{sd}$, an *n-scene* of a scene is the set of cells $\{c = (x_1, x_2, \ldots, x_n, X_{n+1}, \ldots, X_{sd}) \mid c$ is in the scene domain$\}$ together with the associated cell densities. We call X_{n+1} the *location of* the *n-scene* in the $(n + 1)$-scene of the scene, or simply the location of the n-scene. The notion of the n-scene allows us to associate a natural hierarchical order with any given scene and express its sampling scheme as a tree structure. The root of the tree represents the given scene, a node represents an n-scene, and its sons represent the $(n - 1)$-scenes. The leaf nodes represent the 2-scenes (slices) of the scene. Figure 2 shows an example of a 4-D scene with four 3-scenes. The four 3-scenes are composed of two, two, three, and three 2-scenes. Clearly the leaf nodes all appear at the same level of the tree and therefore the tree can be described by a list of the number of sons for each node in the breadth-first order. For the example, this list contains 4, 2, 2, 3, 3. We use this form of tree description to store the number of sons for each node, and except for the root, also the location of the n-scene associated with each node.

IMAGE1. No fixed sampling discipline is assumed for IMAGE1 type of scenes. The location and density values for cells are stored explicitly in the scene data group. (Therefore, message items describing the sampling scheme are not relevant for IMAGE1 type scene data.)

Structure Data

A structure is usually derived from a given scene. We assume that associated with every structure is a *structure domain,* which is usually a subset of the scene domain. For example, if the structure is a 3-D surface, the structure domain is an appropriate 3-D region that encloses the surface; if the structure is a semitransparent volume, the structure domain is the region associated with the volume. We assign a *dimensionality* (*std*) to every structure, which is simply the dimensionality of the structure domain. It is possible that the dimensionality of the scene and of the structure derived from it may be different. For example, the structure may be the 3-D surface of a dynamic object at a particular instance, derived from a 4-D (time-varying) scene. It is not necessary that a structure be always derived from a scene; instead it may be synthesized independently of any underlying scene; for example, an isodose surface in radiation therapy planning or the surface of a prosthesis. In such cases we assume that the structure domain is an appropriate subset of an *sd*-dimensional Euclidean space.

Associated with every structure is a *location* (specified by an *sd*-dimensional vector and an *orientation* (specified by *std* unit vectors each of dimensionality *sd*), which respectively represent the location of the origin and orientation of the axes of the structure coordinate system with respect to the imaging device coordinate system. This information is helpful in determining the location and orientation of the structure with respect to the imaging device during visualization and analysis of the structure. This in turn makes possible the comparative and composite study of objects longitudinally and/or from multimodality scene information.

The *sampling scheme* associated with a structure describes how the structure domain is discretized. This information, specified by the *number of samples* and their *location,* is needed in both visualization and analysis of the structure. We assume that the structure domain is discretized in a fashion similar to how the scene domain is discretized for IMAGE0 type scenes, and so the *number of samples* and their *location* for the structure should be interpreted exactly as the number of n-scenes and their location, described under IMAGE0.

The hierarchical description of a structure as a tree together with the associated terminal (TSE) and nonterminal structure elements (NTSE) (see previous section) is not complete unless the descriptors associated with these

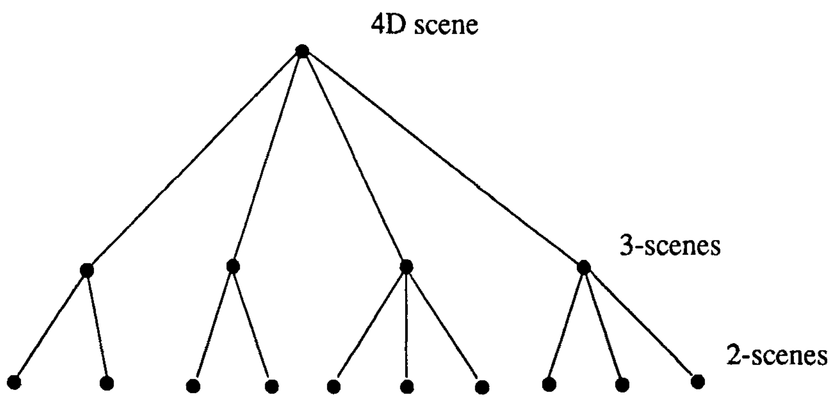

Fig 2. An example illustrating how a 4-D scene is structured using 3- and 2-scenes.

elements are specified. We associate with each terminal and nonterminal structure element an *nct*-tuple TSE and an *ncnt*-tuple NTSE that specify these descriptors. All structure data are stored identically as per the general specification. However, it is the type description that gives a specific interpretation to any given structure data. We describe here some data types; others can be included as required using similar principles.

CURVE0. One method of representing a multidimensional object is by the set of its 2-D boundaries or contours resulting from the intersection of a set of planes with the object. (This structure is commonly used for prosthesis design and manufacture.) Given a scene containing information about the object, the contours are obtained by processing every slice of the scene.[5] A contour in a slice is a boundary of the object in the slice expressed as a sequence of edges of cells (pixels) that forms a closed curve.[5] Figure 3 shows an example. CURVE0 is a structure that represents a set of such contours.

The tree representing the CURVE0 structure of an *std*-dimensional object consists of *std* + 1 levels, as illustrated in Fig 4 for the scene considered in Fig 2. The nonterminal nodes represent the *n*-scenes and the contours, and the terminal nodes represent contour elements. If an *n*-scene does not have any contour specified for it, it is not represented in the tree. The vectors NTSE and TSE associated with the nonterminal and terminal nodes are

$$NTSE = (ns),\ TSE = (y_1, y_2, dir),$$

where ns = the number of sons for the node, y_1, y_2 = the coordinates of the cell to which the

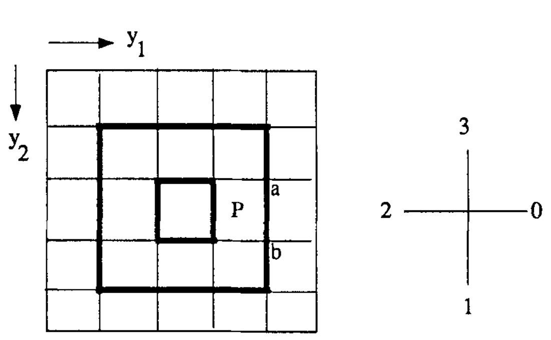

Fig 3. A slice with two contours (bold). A contour element is a line segment such as *ab* that is specified by the cell P of the object to which it belongs and the direction (0) indicating where the line segment lies with respect to the cell.

contour element belongs, and *dir* = the direction of the contour element relative to the cell.

CURVE1, CURVE2, . . . , are type names reserved for other types of boundary curves; for example, for curves that give smoother representations of the contour.

SURFACE0. This structure is used for describing the boundaries of an *std*-dimensional object as sets of (*std* − 1)-dimensional cells, each boundary forming a closed, connected surface in the *std*-dimensional space.[17] A file storing the SURFACE0 structure may contain more than one boundary. In the 3-D case, for example, the cell may be considered to be a cuboid and a boundary is a closed and connected surface made up of faces of the cuboids.[18] A 3-D object may have more than one surface and some or all of these may be stored as a single message.[19]

The hierarchy in the SURFACE0 structure is as shown in Fig 5. Every boundary element (*bel* for short) is represented by the cell in the object to which the bel belongs and a direction (as in Fig 3) indicating where the bel is with respect to the cell. Because there are 2 × *std* possible such directions, the bels may be grouped into 2 × *std* sectors, each sector containing bels of the same orientation. The vectors NTSE and TSE for SURFACE0 are $NTSE = (ns)$, and $TSE = (y_1, y_2, \ldots, y_{std}, n_1, n_2, \ldots, n_{std})$, where ns = the number of sons for the node, $y_1, y_2, \ldots, y_{std}$ = the coordinates of the cell containing the bel, and $n_1, n_2, \ldots, n_{std}$ = the components of a unit surface normal vector associated with the bel.

SURFACE1. This structure is used for describing the 3-D boundaries of a 3-D object as sets of triangular elements (there are several methods that generate such descriptions.[20-22]) As in SURFACE0 type, a file storing a SURFACE1 type boundary may contain more than one boundary. The hierarchical division of the SURFACE1 structure is as in Fig 5 where each bel is a triangle and each sector represents a set of triangles that represent the part of the boundary surface that lies between two successive slices. The descriptions NTSE and TSE are as follows: $NTSE = (ns)$, $TSE = (y_{11}, y_{12}, y_{13}, y_{21}, y_{22}, y_{23}, y_{31}, y_{32}, y_{33}, n_1, n_2, n_3)$, where ns = the number of sons for the node, y_{11}, y_{12}, y_{13} = the coordinates of the first vertex of the bel, y_{21}, y_{22}, y_{23} = the coordinates of the second vertex of the

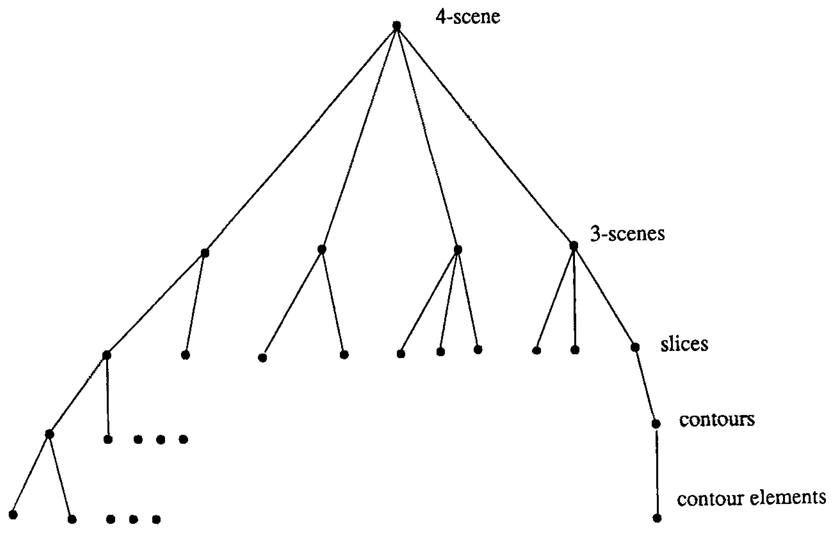

Fig 4. Tree definition of CURVE0 structure for the example in Fig 2.

bel, y_{31}, y_{32}, y_{33} = the coordinates of the third vertex of the bel, n_1, n_2, n_3 = the components of a unit surface normal vector associated with the bel. We reserve SURFACE2, SURFACE3, . . . , type names for other forms of surface description.

SHELL. SHELL is a structure used by processes that allow rendering objects, altering them (for the purpose of simulating surgical manipulations), and measuring them, using both surface[23] and volume rendering techniques.[24-27] The structure is used at present only for 3-D objects (although its concepts generalize to higher dimensions), and therefore we describe here only the 3-D version.

The tree specification for SHELL is as shown in Fig 6. An object here is considered to be a set of cells each with a number of descriptors assigned to it. The cells are organized in the slice and row order. The vectors NTSE and TSE associated with the nodes corresponding to nonterminal and terminal structure elements and their descriptions are as follows: *NTSE* = (*ns*), *TSE* = (*ncode*, y_1, *tt*, n_1, n_2, n_3, *gm*, *op*, *pg*), where

ns	= number of sons for the node
ncode	= a 6-bit number (value 0 to 63) indicating which of the 6-neighbors (immediate neighboring cells in the y_1, y_2, y_3, $-y_1$, $-y_2$, $-y_3$ directions) of the

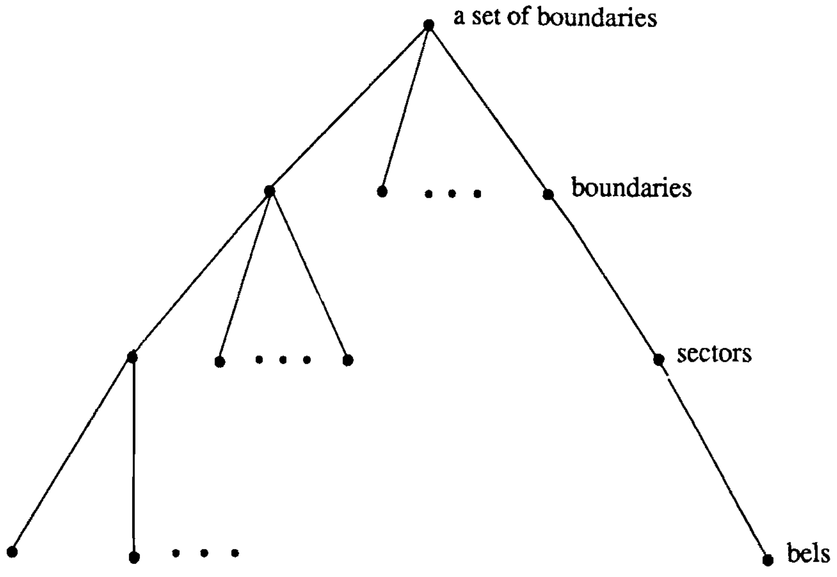

Fig 5. Tree definition for SURFACE0 structure.

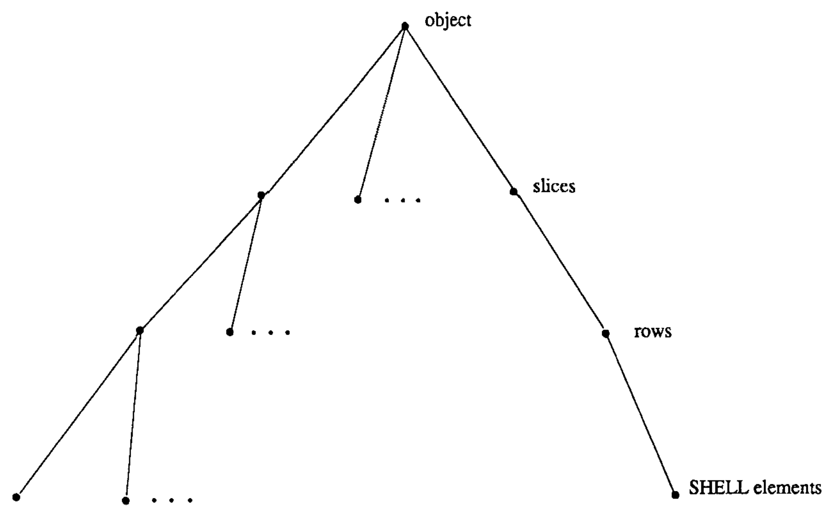

Fig 6. Tree definition for SHELL structure.

cell have an opacity greater than or equal to a fixed high opacity value op_h (bit set to 1) and which have an opacity less than op_h (bit set to 0). If all 6 bits are 1 or if the opacity of the cells is less than a fixed low opacity value op_l, the cell is considered not to be a SHELL element and hence not stored. When $op_h = op_l$, we get a binary SHELL

y_1	$= y_1$ coordinate of the cell;
tt	= tissue type: 0, do not care; 1, air only; 2, air soft tissue mixture; 3, soft tissue only; 4, soft tissue and fat mixture; 5, fat only; 6, fat and bone mixture; 7, bone only
n_1, n_2, n_3	= a vector normal to the object boundary at the center of the cell; n_1 is 0, 1, 2, 3, 4, or 5 depending on which of the six faces of the cell facing respectively the y_1, y_2, y_3, $-y_1$, $-y_2$, $-y_3$ directions is intersected by a line drawn from the center of the cell in the direction of the vector; $n_1 = 6$ means that the normal is not well determined; n_2, n_3 are the coordinates of the point of intersection of the line and the face within the face
gm	= magnitude of the gradient (scaled relative to the minimum and maximum gradient values) of that scene from which the SHELL is derived, evaluated at the cell using an appropriate digital gradient operator
op	= opacity value assigned to the cell;
pg	= tissue percentage (in case of mixture, percentage of the higher density tissue in the descriptor of tt).

It is not necessary to store all of the above information for all rendering techniques. For example, in surface rendering using a binary SHELL,[23] it is not necessary to store *gm, tt, op,* and *pg.* The proposed protocol is general enough to handle a variety of such situations.

Display Data

Each image in a display data set contains a portrayal of some specific aspects of the object information captured in a scene that is ready to be displayed. Sometimes it is possible to parameterize these aspects. We associate a *dimensionality* (*ddd*) with each display data set that constitutes the number of independent parameters characterizing object information captured in the images plus 2 (for the dimensionality of each image). The purpose of identifying the number of independent parameters is to allow the viewer to control the way the values of the parameters are changed in order to decide the order in which images in the display data are presented to the viewer. If such control is not desired or needed, the multiplicity of the parameters can be ignored, resulting in only one independent parameter (and in a dimensionality of three). The only order of presenting the images in the latter case is the natural order in which the images occur in the display data stored as a message. Some examples of display data that may have a dimensionality greater than three are as follows: 1. Each image is a rendition of an object surface for a given view point. If the viewing direction is described by three angles, and if all angles are to be controlled independently, the images comprising the display data can be characterized by three independent parameters. All connected paths in such a 3-D parameter space correspond to valid and possible sequences of displaying the images. Display data dimensionality is thus five. 2. Each image is a rendition as already described, but of a time-varying surface so that time instance becomes the fourth independent parameter. Possible sequences of images now correspond to connected paths in a 4-D parameter space. Display data dimensionality now is six. 3. In addition to these four parameters, we may have up to three additional parameters that specify the changing location of a segment of the object that is separated from the rest of the object and that is being interactively moved by a user. 4. Possible additional parameters are those related to object specification; for example, the object specified by a threshold value of scene density. The display data may constitute the different views of a time-varying object as the threshold value is changed ($ddd = 7$).

Associated with every display data set is a right-handed *display-data coordinate system* $z_1, z_2, \ldots, z_{ddd}$. Associated with every image of the display data is a *parameter vector* that has ddd-2 components, each of which corresponds to one of the independent parameters that characterize the image. Associated with every pixel of the image is an *intensity vector* that stores the values associated with the pixels.

MOVIE0. This is a particular type of display data that represents cine sequences of surface and volume renditions. The image intensity vector IV is 1-D or 3-D, and the parameter vector PV may contain any (finite) number of components. The meaning associated with the components is not fixed and is left to the user to determine (see next section). The examples given earlier indicate some possible meanings. $IV = (iv)$ or (iv_1, iv_2, iv_3), $PV = (p_1, p_2, \ldots, p_{ddd-2})$, where $iv1$ = red component, $iv2$ = green component, $iv3$ = blue component; when IV is scalar-valued, the value represents pixel gray value.

FORMAT DESCRIPTION

A *message* is a set of related data items and each data file contains one message. Each item in a message is assigned a group number and an element number and is completely represented in the file by storing its GROUP#, ELEMENT#, LENGTH, and VALUE, where LENGTH is the number of bytes of storage required for the item and VALUE is its actual value. A file thus contains a sequence of such quadruples. If the message is a 2-D scene, for example, some message items (quadruples) store information related to the description of the scene such as its dimension, size, resolution, and so on, and the quadruple associated with group 7FE0 (Table 1) stores the actual pixel density values. The even-numbered groups are called *standard groups* and are reserved by the ACR-NEMA standards committee. The odd-numbered groups are called *shadow groups* and may be used for private purposes. All type-1 and type-1D items from standard groups must be present in the message.

The detailed format of storing the items in a message is presented in Table 1. All entries under Group, Element and Length are hexadecimal numbers. Groups numbered 0009, 0029, 002B, 002D, and 8001 do not exist in the 2-D ACR-NEMA standards[15] and have been created to accommodate multidimensional scene, structure, and display data. The meaning of GROUP:0000, ELEMENT:0800, GROUP:0008, ELEMENT:0041, and of GROUP:7FE0 have also been broadened. In addition, we have created a new value representation (VR) type called generalized numeric (GN) that allows storing sets of mixed integer and floating point numbers for message items, where the integers may be bit-packed in any specified fashion and the floating point numbers may be single or double precision.

Messages representing data of all types contain message items up to and including GROUP:0028. Beyond GROUP:0028, message items are included based on their relevance to the data type. Message items relevant to each of the scene, structure, and display data types are as follows: scene = groups up to GROUP:0028 and GROUP:0029, GROUP:7FE0; structure = groups up to GROUP:0028 and GROUP:002B,

Table 1. Detailed Data Format Specification

Group	Element	Length		Value	VR	VM	Type	Description
Command information group								
0000	0000	0004	0000	—	BD	S	1	Group length (an even number of bytes from the end of the value field to the beginning of the next group)
0000	0001	0004	0000	—	BD	S	1	Message length (an even number of bytes from the end of the value field to the beginning of the next message)
0000	0010	—	—	—	AT	S	1	Recognition code (used for identifying the version number of this protocol)
0000	0100	0002	0000	—	BI	S	1	Command field
0000	0110	0002	0000	—	BI	S	1	Message ID
0000	0120	0002	0000	—	BI	S	1	Message ID being responded to
0000	0200	—	—	—	AT	S	1	Initiator
0000	0300	—	—	—	AT	S	1D	Receiver
0000	0400	—	—	—	AT	S	1	FIND location
0000	0600	—	—	—	AT	S	1	MOVE
0000	0700	0002	0000	—	BI	S	1D	Priority
0000	0800	0002	0000	—	BI	S	1	Data set type:
								Scene data:
								IMAGE0 = 0
								IMAGE1 = 1
								Structure data:
								CURVE0 = 100
								SURFACE0 = 110
								SURFACE1 = 111
								SHELL = 120
								Display data:
								MOVIE0 = 200
								Other data:
								GRAPHICS = 2 (see [15])
								TEXT = 3 (see [15])
0000	0850	0002	0000	—	BI	S	1D	Number of matches
0000	0860	0002	0000	—	BI	S	1D	Response sequence number
0000	0900	0002	0000	—	BI	S	1	Status
0000	4000	—	—	—	AT	S	1D	DIALOG receiver
0000	4010	—	—	—	AT	S	1D	Terminal type
Identification information group								
0008	0000	0004	0000	—	BD	S	1	Group length
0008	0001	0004	0000	—	BD	S	1	Message length (same as GROUP:0000, ELEMENT:0001)
0008	0010	—	—	—	AT	S	1	Recognition code (used for identifying the version number of this protocol)
0008	0020	000A	0000	—	AT	S	2	Study date (the format is yyyy.mm.dd; eg, 1988.07.18)
0008	0030	0008	0000	—	AT	S	2	Study time (the format is hh:mm:ss; eg, 10:05:03 (24 hour time))
0008	0040	0002	0000	—	BI	S	1	Data set type (same as GROUP:0000, ELEMENT:0800)
0008	0041	—	—	—	AT	S	3	Data set subtype
0008	0060	0002	0000	—	AT	S	2	Modality (CT, NM, MR, DS, DR, US, OT)
0008	0070	—	—	—	AT	S	2	Manufacturer
0008	0080	—	—	—	AT	S	2	Institution ID
0008	0090	—	—	—	AT	M	2	Referring physician
0008	1000	—	—	—	AT	S	3	Network ID
0008	1010	—	—	—	AT	S	2	Station ID

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length	Value	VR	VM	Type	Description
Identification information group: (continued)							
0008	1030	—	—	—	AT	S	3	Procedure description
0008	1040	—	—	—	AT	S	3	Institutional department
0008	1050	—	—	—	AT	M	3	Attending physician
0008	1060	—	—	—	AT	M	3	Radiologist
0008	1070	—	—	—	AT	M	3	Operator identification
0008	1080	—	—	—	AT	M	3	Admitting diagnosis
0008	1090	—	—	—	AT	S	3	Manufacturer model
0008	4000	—	—	—	AT	M	3	Comments
General information group							
0009	0000	0004	0000	—	BD	S	1	Group length
0009	8000	—	—	—	AT	S	3	Name of file containing this message
0009	8010	—	—	—	AT	S	3	Name of file that gave rise to this file
0009	8020	—	—	—	AT	S	3	Description of data in file
0009	8030	—	—	—	AT	S	3	Scratch pad (to store any user-specified information, eg, measurements made on the data, details of modifications made to the data, etc)
Patient information group							
0010	0000	0004	0000	—	BD	S	1	Group length
0010	0010	—	—	—	AT	S	2	Patient name
0010	0020	—	—	—	AT	S	2	Patient ID
0010	0030	—	—	—	AT	S	2	Patient birthdate
0010	0040	—	—	—	AT	S	2	Patient sex
0010	1000	—	—	—	AT	M	3	Other patient IDs
0010	1001	—	—	—	AT	M	3	Other patient name
0010	1005	—	—	—	AT	S	3	Patient's maiden name
0010	1010	—	—	—	AT	S	3	Patient age
0010	1020	—	—	—	AT	S	3	Patient size
0010	1030	—	—	—	AT	S	3	Patient weight
0010	1040	—	—	—	AT	S	3	Patient address
0010	1050	—	—	—	AT	M	3	Insurance plan ID
0010	1060	—	—	—	AT	S	3	Patient's mother's maiden name
0010	4000	—	—	—	AT	M	3	Comments
Acquisition information group							
0018	0000	0004	0000	—	BD	S	1	Group length
0018	0010	—	—	—	AT	M	2D	Contrast/bolus agent
0018	0020	—	—	—	AT	M	2	Scanning sequence
0018	0030	—	—	—	AT	M	2	Radionuclide
0018	0040	—	—	—	AN	S	2D	Cine rate
0018	0050	—	—	—	AN	S	2	Slice thickness
0018	0060	—	—	—	AN	M	2	KVP
0018	0070	—	—	—	AN	S	2	Counts accumulated
0018	0080	—	—	—	AN	S	2	Repetition time
0018	0081	—	—	—	AN	S	2	Echo time
0018	0082	—	—	—	AN	S	2	Inversion time
0018	0083	—	—	—	AN	S	2	Number of averages
0018	0084	—	—	—	AN	S	2	Imaging frequency
0018	0085	—	—	—	AT	S	2	Imaged nucleus
0018	1000	—	—	—	AT	S	3	Device serial number
0018	1010	—	—	—	AT	S	3	Film scanner ID

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length		Value	VR	VM	Type	Description
Acquisition information group: (continued)								
0018	1020	—	—	—	AT	S	3	Software version
0018	1030	—	—	—	AT	S	3	Protocol
0018	1040	—	—	—	AT	M	3	Contrast/bolus route
0018	1041	—	—	—	AN	M	3	Contrast/bolus volume
0018	1042	—	—	—	AT	M	3	Contrast/bolus start time
0018	1043	—	—	—	AT	M	3	Contrast/bolus stop time
0018	1044	—	—	—	AN	M	3	Contrast/bolus total dose
0018	1050	—	—	—	AN	M	3	Spatial resolution
0018	1060	—	—	—	AN	S	3	Trigger time
0018	1070	—	—	—	AT	M	3	Radionuclide route
0018	1071	—	—	—	AN	M	3	Radionuclide volume
0018	1072	—	—	—	AT	M	3	Radionuclide start time
0018	1073	—	—	—	AT	M	3	Radionuclide stop time
0018	1074	—	—	—	AN	M	3	Radionuclide total dose
0018	1100	—	—	—	AN	S	3	Reconstruction diameter
0018	1110	—	—	—	AN	S	3	Distance source to detector
0018	1111	—	—	—	AN	S	3	Distance source to patient
0018	1120	—	—	—	AN	S	3	Gantry tilt
0018	1130	—	—	—	AN	S	3	Table height
0018	1140	—	—	—	AT	S	3	Rotation direction
0018	1150	—	—	—	AN	S	3	Exposure time
0018	1151	—	—	—	AN	S	3	Exposure rate
0018	1152	—	—	—	AN	S	3	Exposure
0018	1160	—	—	—	AT	M	3	Filter type
0018	1170	—	—	—	AN	S	3	Generator power
0018	1180	—	—	—	AT	M	3	Collimator/grid
0018	1190	—	—	—	AN	M	3	Focal spot
0018	1200	—	—	—	AT	M	3	Date of last calibration
0018	1201	—	—	—	AT	M	3	Time of last calibration
0018	1210	—	—	—	AT	M	3	Convolution kernel
0018	1240	—	—	—	AN	M	3	Upper/lower pixel values
0018	1242	—	—	—	AN	S	3	Termination time
0018	1243	—	—	—	AN	S	3	Count rate
0018	1250	—	—	—	AT	S	3	Receiving coil
0018	1251	—	—	—	AT	S	3	Transmitting coil
0018	1260	—	—	—	AT	S	3	Screen type
0018	1261	—	—	—	AT	S	3	Phosphor type
0018	4000	—	—	—	AT	M	3	Comments
Relationship information group								
0020	0000	0004	0000	—	BD	S	1	Group length
0020	0010	—	—	—	AN	S	2	Study (exam number)
0020	0011	—	—	—	AN	S	2D	Series
0020	0012	—	—	—	AN	S	2D	Acquisition
0020	0013	—	—	—	AN	S	2D	Image number
0020	0020	—	—	—	AT	M	2	Patient orientation
0020	0030	—	—	—	AN	M	2	Image position
0020	0035	—	—	—	AN	M	2D	Image orientation
0020	0050	—	—	—	AN	S	2	Location
0020	0060	—	—	—	AT	S	2D	Laterality
0020	0070	—	—	—	AT	S	2D	Image geometry type
0020	0080	—	—	—	AT	M	2D	Masking image
0020	1000	—	—	—	AN	S	3	No. of series in this study
0020	1001	—	—	—	AN	S	3	No. of acquisitions in this series
0020	1002	—	—	—	AN	S	3	No. of images from this acquisition

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length		Value	VR	VM	Type	Description
Relationship information group: (continued)								
0020	1020	—	—	—	AT	M	3	Reference
0020	1040	—	—	—	AT	S	3	Position reference indicator
0020	1041	—	—	—	AN	S	3	Slice location
0020	1070	—	—	—	AN	M	3	Other study numbers
0020	3100 to 31FF	—	—	—	AT	M	2D	Source image IDs
0020	3401	—	—	—	AT	M	2D	Modifying device ID
0020	3402	—	—	—	AT	M	2D	Modified image ID
0020	3403	—	—	—	AT	M	2D	Modified image date
0020	3404	—	—	—	AT	M	2D	Modifying device manufacturer
0020	3405	—	—	—	AT	M	2D	Modified imaging time
0020	3406	—	—	—	AT	M	2D	Modified image description
0020	4000	—	—	—	AT	M	3	Comments
Image presentation information group								
0028	0000	0004	0000	—	BD	S	1	Group length
0028	0005	0002	0000	—	BI	S	1D	Image dimensions
0028	0010	0002	0000	—	BI	S	1	No. of rows
0028	0011	0002	0000	—	BI	S	1	No. of columns
0028	0030	—	—	—	AN	M	2	Pixel size
0028	0040	—	—	—	AT	S	1D	Image format
0028	0050	—	—	—	AT	M	2D	Manipulated image
0028	0060	—	—	—	AT	S	1D	Compression code
0028	0100	0002	0000	—	BI	S	1D	Bits allocated
0028	0101	0002	0000	—	BI	S	1D	Bits stored
0028	0102	0002	0000	—	BI	S	1D	High bit
0028	0104	0002	0000	—	BI	S	2D	Smallest pixel value
0028	0105	0002	0000	—	BI	S	2D	Largest pixel value
0028	0200	0002	0000	—	BI	S	1D	Image location
0028	1050	—	—	—	AN	M	3	Window center
0028	1051	—	—	—	AN	M	3	Window width
0028	1052	—	—	—	AN	M	3	Rescale intercept
0028	1053	—	—	—	AN	M	3	Rescale slope
0028	1080	—	—	—	AT	S	3	Gray scale
0028	1100	—	—	—	BI	M	2D	Lookup table descriptors; gray [no. of entries in the table (net)\starting pixel value (spv)\number of bits for entries]
0028	1101	—	—	—	BI	M	2D	Lookup table descriptors; red (as for gray)
0028	1102	—	—	—	BI	M	2D	Lookup table descriptors; green (as for gray)
0028	1103	—	—	—	BI	M	2D	Lookup table descriptors; blue (as for gray)
0028	1200	—	—	—	BI	M	2D	Lookup data; gray (mapped value 1\mapped value 2\ . . .\mapped value net; pixel values to be mapped are assumed to be spv, spv + 1, . . . , spv + net − 1; each mapped value is stored in 16 bits)
0028	1201	—	—	—	BI	M	2D	Lookup data; red (as for gray)
0028	1202	—	—	—	BI	M	2D	Lookup data; green (as for gray)
0028	1203	—	—	—	BI	M	2D	Lookup data; blue (as for gray)
0028	4000	—	—	—	AT	M	3	Comments
Scene-related information group*								
0029	0000	0004	0000	—	BD	S	1	Group length
0029	8000	0002	0000	—	BI	S	1	Scene dimension (sd)

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length	Value	VR	VM	Type	Description
Scene-related information group: (continued)							
0029	8010	—	—	—	AN	M	1D Location and orientation of the scene domain with respect to a fixed imaging device coordinate system: specified by (sd + 1) vectors $X_0, \ldots, X_{sd}$. X_0: coordinates of the origin of the scene coordinate system. $X_1, \ldots, X_{sd}$: unit vectors along each of its sd principal directions
0029	8015	—	—	—	AT	M	2D Labels associated with the axes
0029	8020	—	—	—	BI	M	1D Unit of measurement in the sd directions: 0, km; 1, m; 2, cm; 3, mm; 4, μm; 5, s; 6, ms; 7, μs
0029	8030	0002	0000	—	BI	S	1 No. of density values associated with each cell (nd)
0029	8040	—	—	—	AN	M	1D Smallest density value (when density is vector valued, the smallest of each component in the scene)
0029	8050	—	—	—	AN	M	1D Largest density value (when density is vector valued, the largest of each component in the scene)
0029	8060	0002	0000	—	BI	S	1 Density storage scheme: number of integers in the density vector (ni). (The rest nd—ni are assumed to be floating-point numbers)
0029	8070	—	—	—	BI	M	1D Density storage scheme: whether each integer component is signed or unsigned: sousl\...\sousni\, where each entry indicates whether each integer component of the density vector is signed (1) or unsigned (0)
0029	8080	0002	0000	—	BI	S	1 Density storage scheme: total bits required for the density vector (nbi)
0029	8090	—	—	—	BI	M	1 Density storage scheme: lbit1\rbit1\...\lbitnd\rbitnd\. Within the bit field of nbi bits for the density vector, the bits are numbered 0, 1, ..., nbi-1 from left to right (most significant to least significant) and the bit field for each component is specified by a left bit (lbit) and a right bit (rbit). Thus, for every component lbit ≤ rbit. The lbit, rbit pair for a floating point number indicates whether the number is of single or double precision (see the definition of value representation GN in Section 2). The density vector itself is stored as integer1\ integer2\ ...\integerni\fp1\...\fp(nd-ni) within the nbi bits bit field and the order of the bit fields is preserved in the file storing the message.
0029	8095	0004	0000	—	BI	M	1 Size of each slice: null (ie, LENGTH = 0) for IMAGE1 type; for IMAGE0 type this element stores no. of columns\no. of rows\in the slice
0029	80A0	—	—	—	BI	M	1 Sampling scheme (no. of n-scenes): null (ie, LENGTH = 0) for IMAGE1 type. For IMAGE0 type, this element stores, for each n, 2 < n ≤ sd, the number of (n − 1)-scenes for each n-scene. These numbers are stored in the order ns0\ns00\ns01\ns02\... \ns000\ns001\....(See IMAGE0)

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length		Value	VR	VM	Type	Description
Scene-related information group: (continued)								
0029	80A5	—	—	—	AN	M	1	Pixel size in the x_1 and x_2 directions: psizex1\psizex2
0029	80B0	—	—	—	AN	M	1	Sampling scheme (location of n-scenes): null (ie, LENGTH = 0) for IMAGE1 type. For IMAGE0 type, this element stores, for each n, $2 < n \leq$ sd, the location of each (n − 1)-scene of each n-scene in the order loc00\loc01\loc02\. . . .\loc000\loc001\. . . . (See IMAGE0)
0029	80C0	—	—	—	AT	M	3	Description of processing operations on this scene and their parameters: type of processing (filtering, segmentation, interpolation, classification, masking, reslicing, histogram transforms, ROI), methods, parameters of the method
Structure-related information group†								
002B	0000	0004	0000	—	BD	S	1	Group length
002B	8000	0002	0000	—	BI	S	1	Structure dimension (std)
002B	8010	—	—	—	AN	M	1D	Location and orientation of the structure coordinate system with respect to the imaging device coordinate system: specified by std + 1 vectors $Y_0, Y_1, \ldots, Y_{std}$. Y_0: coordinates of the origin of the structure coordinate system. $Y_1, \ldots, Y_{std}$: unit vectors along each of its std principal directions
002B	8015	—	—	—	AT	M	2D	Labels associated with the axes
002B	8020	—	—	—	BI	M	1D	Unit of measurement in the std directions: 0, km; 1, m; 2, cm; 3, mm; 4, μm; 5, s; 6, ms; 7, μs
002B	8030	0004	0000	—	BD	S	1	No. of terminal structure elements
002B	8040	0004	0000	—	BD	S	1	No. of nonterminal structure elements
002B	8050	0002	0000	—	BI	S	1	No. of components in the vector TSE assigned to each terminal structure element (nct)
002B	8060	0002	0000	—	BI	S	1	No. of components in the vector NTSE assigned to each nonterminal structure element (ncnt)
002B	8070	—	—	—	AN	M	1D	Smallest value of each component of TSE in the structure
002B	8080	—	—	—	AN	M	1D	Largest value of each component of TSE in the structure
002B	8090	0002	0000	—	BI	S	1	Storage for terminal structure elements: no. of integers in TSE (nit). (The rest of the nct-nit components are floating point numbers)
002B	80A0	—	—	—	BI	M	1D	Whether each integer component of TSE is signed or unsigned: sous1\sous2\. . .\sousnit\, where each entry indicates whether each integer component of TSE is signed (1) or unsigned (0)
002B	80B0	0002	0000	—	BI	S	1	Storage scheme for terminal structure elements: total bits required for TSE (nbit)

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length	Value	VR	VM	Type	Description
Structure-related information group: (continued)							
002B	80C0	—	—	—	BI	M	1	Storage scheme for TSE: lbit1\rbit1\. . .\lbitnct\rbitnct\. Description is similar to that for GROUP:0029, ELEMENT:8090. If lbit > rbit, it means that the component of TSE associated with this field is not stored
002B	80D0	0002	0000	—	BI	S	1	Storage scheme for nonterminal structure elements: no. of integers in NTSE (nint). (The rest of the ncnt-nint components are floating point numbers)
002B	80E0	—	—	—	BI	M	1D	Whether each integer component of NTSE is signed or unsigned: sous1\sous2\. . .\sousnint\, where each entry indicates whether each integer component of NTSE is signed (1) or unsigned (0)
002B	80F0	0002	0000	—	BI	S	1	Storage scheme for nonterminal structure elements: no. of bits required for NTSE (nbint)
002B	8100	—	—	—	BI	M	1	Storage scheme for NTSE: lbit1\rbit1\. . .\lbitncnt\rbitncnt\. Description is similar to that for GROUP:0029, ELEMENT:8090
002B	8110	—	—	—	BI	M	1	Sampling scheme for the structure: this element stores, for each n, 2 < n ≤ std, the number of samples in each dimension. These numbers are stored in the order ns0\ns00\ns01\ns02\. . .\ns000\ns001\. . . . (see previous section)
002B	8120	—	—	—	AN	M	1	Sample size in the y_1 and y_2 directions: ssizey1\ssizey2
002B	8130	—	—	—	AN	M	1	Sampling scheme for the structure: This element stores, for each n, 2 < n ≤ std, the sample location in the order loc00\loc01\loc02\. . .\loc000\loc001\. . . (see previous section)
002B	8140	—	—	—	AN	M	2	Quantitative information about the structure: minimum and maximum coordinates for structure elements (with respect to the structure coordinate system): miny1\miny2\. . .\minystd\maxy1\maxy2\. . .\maxystd \. This element specifies the extent of the structure within the structure coordinate system
002B	8150	—	—	—	AN	S	3	Quantitative information about the structure: volume
002B	8160	—	—	—	AN	S	3	Quantitative information about the structure: surface area
002B	8170	—	—	—	AN	S	3	Quantitative information about the structure: rate of change of volume
002B	8180	—	—	—	AT	M	3	Description of structure extraction method and its parameters

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length		Value	VR	VM	Type	Description
Display-related information group‡								
002D	0000	0004	0000	—	BD	S	1	Group length
002D	8000	0002	0000	—	BI	S	1	Dimensionality of the display data (ddd)
002D	8010	0004	0000	—	BI	M	1D	Unit of measurement for the pixel size: 0, km; 1, m; 2, cm; 3, mm; 4, μm; 5, s; 6, ms; 7, μs
002D	8020	0002	0000	—	BI	S	1	Intensity storage scheme: no. of elements in the intensity vector associated with the pixels (neiv)
002D	8030	—	—	—	AN	M	1D	Smallest value of each component of the intensity vector in the display
002D	8040	—	—	—	AN	M	1D	Largest value of each component of the intensity vector in the display
002D	8050	0002	0000	—	BI	S	1	Intensity storage scheme: no. of integer-valued elements in the intensity vector (niiv). (The rest of the neiv−niiv elements are assumed to be floating point numbers)
002D	8060	—	—	—	BI	M	1D	Whether each of the integer-valued elements in the intensity vector is signed or unsigned: sous1\...\sousniiv\, where each entry indicates whether each integer component of the intensity vector is signed (1) or unsigned (0)
002D	8070	0002	0000	—	BI	S	1	Intensity storage scheme: total bits required for the intensity vector (nbiiv)
002D	8080	—	—	—	BI	M	1	Intensity storage scheme: lbit1\rbit1\...\lbitneiv\rbitneiv\. This element specifies the bit field for each element of the intensity vector via a left-bit, right-bit pair of numbers; see GROUP: 0029, ELEMENT:8090
002D	8090	0002	0000	—	BI	S	1	Total images in the display (tni)
002D	80A0	0004	0000	—	BI	M	1	Size of the image: no. of columns\no. of rows\
002D	80B0	—	—	—	AN	M	1D	Pixel size in the z_1 and z_2 directions
002D	80C0	—	—	—	AT	M	2	Specification of the meaning of each component of PV: meaning1\meaning2\...\meaning(ddd-2)
002D	80D0	—	—	—	BI	M	2	Characteristics of the images: This element stores the parameter vectors that describe the characteristics of the images in the display: *PV*1\PV2\\....\PVtni\
002D	80E0	—	—	—	AT	M	3	Description of visualization method and its parameters for each object [eg, volume rendering or surface rendering; ray-casting or projection; shading method (scene-space, view space, object space); shading parameters (diffuse parameters, specular parameters, ambient); antialiasing method; color of the object; etc]
Scene data group								
7FE0	0000	0004	0000	—	BD	S	1	Group length: A value FFFF FFFF signals that the number of bytes required for the cell density values ≥ FFFF FFFF and that data might be stored in subsequent odd-numbered groups 7FE1, 7FE3, . . . , following cell density values in this group

Table 1. Detailed Data Format Specification (Cont'd)

Group	Element	Length	Value	VR	VM	Type	Description
IMAGE0 type							
7FE0	0010	—	—	—	GN	M	1	Cell densities: stored as per density storage scheme, as a sequence of *nd*-tuples CD1\CD2\. . . , where CDi contains the densities associated with the cell numbered *i*. The cells are numbered consecutively changing the coordinates in the following order: $x_1, x_2, \ldots, x_{sd}$ (ie, in the row-by-row, slice-by-slice, etc order)
7FE0	0010	—	—	—	GN	M	1	Cell specification (locations and densities): stored as per density storage scheme, as a sequence of pairs LOC1\CD1\LOC2\CD2\. . . , where LOCi is an sd-tuple indicating the location of cell *i* and CD*i* is an nd-tuple storing the densities of cell *i*
Structure data group							
8001	0000	0004	0000	—	BD	S	1	Group length: A value of FFFF FFFF signals that the no. of bytes required for structure data is ≥ FFFF FFFF and that data might be stored in subsequent odd-numbered groups 8003, 8005, . . . , following data in this group
8001	8000	—	—	—	GN	M	1	Structure data: The vectors (NTSE or TSE) assigned to the nodes of the tree representing the structure are stored in the breadth-first order of the nodes (see Section 3) vector for node 0\vector for node 00\vector for node 01\. . .\vector for node 000\. . . . Note that, because of the nature of the tree, for structures of type CURVE0, SURFACE0, and SHELL, the vectors NTSE come first in this element, which are then followed by the vectors TSE. For other general tree structures this order may not occur
Display data group							
8021	0000	0004	0000	—	BD	S	1	Group length: A value of FFFF FFFF signals that the no. of bytes required for display data is ≥ FFFF FFFF and that data might be stored in subsequent odd-numbered groups 8023, 8025, . . . , following data in this group
8021	8000	—	—	—	GN	M	1	Display data: The intensity vectors (IV) are stored as per intensity storage scheme as a sequence of neiv tuples IV1\IV2\. . . where IV*i* contains the intensities associated with the pixel numbered *i*. The pixels are numbered in the row-by-row order

*See Scene Data for definition of terms.
†See Structure Data.
‡See Display Data.

GROUP:8001; display = groups up to GROUP: 0028 and GROUP:002D, GROUP:8021.

LIMITATIONS AND EFFICIENCY

The existing ACR-NEMA standards specification[15] has two important limitations. First, the byte ordering specified by the standards for value representation methods denoted BD, AN, and AT is not natural for machines that consider the bytes to be in a monotonically decreasing order of significance from left to right. This calls for rearrangement of bytes whenever a standard message is to be interpreted by such machines or when they must create the message in the standard form from an internal representation. Multidimensional imaging systems can handle this conflict by gathering machine-specific byte-order information (at the time of installation of the system) and by automatically reordering bytes when needed. The loss of efficiency incurred in this process is not significant because types BD, AN, and AT occur only in the descriptive part (header) of messages.

The second limitation comes from the requirement that all floating point numbers be stored in ASCII numeric form. Although in many situations floating point numbers can be replaced by integers without much loss of information, fulfillment of this condition may not be guaranteed in general. The new value representation method called GN suggested in this protocol is an effective solution to this limitation.

Because we wanted this specification to be compatible with the existing ACR-NEMA standards, some of the groups we have created have information of somewhat similar nature to that occurring in standard groups. The specification can be made more efficient by merging relevant groups or elements from such groups. For example, image presentation and scene-related information groups can be merged into a single group to eliminate some of the repetitions.

Table 2. Storage Requirement for Some Typical Data Sets

	Header	Data
IMAGE0	1 KB	9 MB
CURVE0	2 KB	0.3 MB
SURFACE0	0.6 KB	2.7 MB
SHELL	2.5 KB	1.2 MB

Since descriptive information (headers) pertaining to scene and structure data is stored on a need basis, the additional storage required owing to the generality of the specification can be kept to a minimum. We emphasize again that for a given data type only those message items that are relevant to the data type are stored. The storage requirement figures determined for a typical data set (Table 2) give some idea of the cost of storage for the header part of the data (groups up to 002D) and for the data part (groups beyond 002D).

CONCLUSIONS

As multidimensional image processing, visualization, and analysis become more common, standard means of representing and storing image and nonimage data will be needed for convenient exchange of information, and for development of software environments that are application, data, machine, and approach independent. In this article, we have described an extension to and a generalization of the 2-D ACR-NEMA standards to meet these requirements. The generalized specification is actively being used in a software environment being developed by us, called 3DVIEWNIX, for multidimensional biomedical image display, processing, and analysis.

ACKNOWLEDGMENT

We thank Mary A. Blue for typing the manuscript.

REFERENCES

1. Stark DD, Bradley WG: Magnetic Resonance Imaging. St. Louis, MO, Mosby, 1988, pp 172-174

2. Robb RA: High-speed three-dimensional x-ray computed tomography. Proc IEEE 71:308-319, 1983

3. Gadian DG: NMR and Its Application to Living Systems. Oxford, England, Oxford University Press, 1982

4. Principe JC, Yu FS, Reid SA: Display of EEG chaotic dynamics, in Proceedings, First International Conference on Visualization in Biomedical Computing. Atlanta, GA, 1990. Los Alamitos, CA, IEEE Computer Society, 1990, pp 346-351

5. Udupa JK, Herman GT (eds): 3D Imaging in Medicine. Boca Raton, FL, CRC 1991

6. Axel L, Dougherty L: MR imaging of motion with spatial modulation of magnetization. Radiology 171:841-845, 1989

7. Axel L, Goncalves R, Bloomgarden D: Two-dimensional analysis and functional imaging of regional heart wall

motion with magnetic resonance imaging. Radiology (in press)

8. Tag Image File Format—Rev.5.0, An Aldus Corporation (Seattle, WA)/Microsoft Corporation (Redmond, WA) Technical Memorandum, 1987

9. Thomas SG: Design of the Utah RLE format. Technical Report 86-15, Alpha-1 Project, Computer Science Department, University of Utah, Salt Lake City, November 1986

10. Macintosh technical notes No. 21 (quickdraw's internal picture definition), No. 41 (drawing to an offscreen bitmap), No. 120 (drawing to an offscreen pixmap), and No. 154 (displaying large pict files), Apple Computer, Inc, Cupertino, CA

11. Treinish LA: SIGGRAPH '90 workshop report: Data structures and access software for scientific visualization. Computer Graphics, 25:104-118, 1991

12. Hung HM, Udupa JK, Goncalves R, et al: The 3DVIEWNIX software system—data-, graphics-, and process-interface functions—version 1.0. Technical Report MIPG178, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA, January 1991

13. Thompson KL, Ritchie DM: The UNIX time-sharing system. Communications ACM 17:365-375, 1974

14. Scheifler RW, Gettys J, Newman R: X Window System C Library and Protocol Reference. Bradford, MA, Digital, 1988

15. ACR-NEMA: Digital imaging and communications. ACR-NEMA Standards Publication No. 300-1985, Washington, DC, National Electrical Manufacturer's Association, 1985

16. Palmer JF, Morse SP: The 8087 Primer. New York, NY, Wiley 1984, pp 12-28

17. Udupa JK: A unified theory of objects and their boundaries in multidimensional digital images, in Jaffee CC, Lemke HU, Rhodes ML, et al (eds): Proceedings of Computer Assisted Radiology, CAR'87. Berlin, Germany, July 1-4, 1987, pp 779-784

18. Udupa JK, Hung HM, Chuang KS: Surface and volume rendering in 3D imaging: A comparison. J Digital Imaging 4:159-168, 1991

19. Udupa JK, Ajjanagadde VG: Boundary and object labelling in three-dimensional images. Comput Vision Graphics Image Processing 51:355-369, 1990

20. Cline HE, Lorenson WE, Ludke S, et al: Two algorithms for the three-dimensional reconstruction of tomograms. Med Physics 15:320-327, 1987

21. Fuchs H, Kedem ZM, Uselton SP: Optimal surface reconstruction for planar contours. Communications ACM 20:693-702, 1977

22. Boissonnat JD: Shape reconstruction from planar cross sections. Comput Vision Graphics Image Processing 44:1-29, 1988

23. Udupa JK, Odhner D: Interactive surgical planning: high-speed object rendition and manipulation without specialized hardware, in: Proceedings of the First International Conference on Visualization in Biomedical Computing. Atlanta, GA, 1990. Los Alamitos, CA, IEEE Computer Society, 1990, pp 330-336

24. Levoy M: Display of surfaces from volume data. IEEE Comput Graphics Applications 8:29-37, 1988

25. Drebin RA, Carpenter L, Hanrahan P: Volume rendering. Comput Graphics 22:65-74, 1988

26. Kaufman A: A Tutorial on Volume Visualization. Los Alamitos, CA, IEEE Computer Society Press, 1990

27. Hoehne KH, Bomans M, Pommert A, et al: 3D-visualization of tomographic volume data using the generalized voxel model. Visual Comput 6:28-37, 1990