Cytometry 11:561–569 (1990)

SPECIAL REPORT

Proposed Standard for Image Cytometry Data Files[1]

Phillip Dean,[2] Laura Mascio, David Ow, Damir Sudar, and James Mullikin

Biomedical Sciences Division, Lawrence Livermore National Laboratory, Livermore, California 94551 (P.D., L.M., D.O.); Department of Laboratory Medicine, University of California, San Francisco, California 94143 (D.S.); Pattern Recognition Group, Delft University of Technology, Delft, The Netherlands (J.M.)

Received for publication August 31, 1989; accepted March 13, 1990

A number of different types of computers running a variety of operating systems are presently used for the collection and analysis of image cytometry data. In order to facilitate the development of sharable data analysis programs, to allow for the transport of image cytometry data from one installation to another, and to provide a uniform and controlled means for including textual information in data files, this document describes a data storage format that is proposed as a standard for use in image cytometry. In this standard, data from an image measurement are stored in a minimum of two files. One file is written in ASCII to include information about the way the image data are written and optionally, information about the sample, experiment, equipment, etc. The image data are written separately into a binary file. This standard is proposed with the intention that it will be used internationally for the storage and handling of biomedical image cytometry data. The method of data storage described in this paper is similar to those methods published in *American Association of Physicists in Medicine (AAPM) Report Number 10* and in *ACR-NEMA Standards Publication Number 300-1985.*

Key terms: Analysis, processing, storage, database

There are many standards for the storage of data on computers: e.g., the DIF format (Lotus Development Corp., Cambridge, MA) used to transfer data among programs on personal computers; the TIFF (tag image file format) for storing bit-mapped images, developed especially for gray-scale data; and the PICT format developed by Apple Computer to handle graphics on the Macintosh. These formats were not selected for our application due to their hardware dependence and their lack of provision for the inclusion of related information. Other formats have been proposed for more general use. One is the American Association of Physicists in Medicine (AAPM) Standard for Digital Image Interchange (2). This standard was developed for the medical imaging community, particularly for network transfer of image data. It uses a header file, written as ASCII character strings, to contain non-pixel information. A second file contains the image data written in binary format. Three other groups have proposed similar methods of storing image data: CART, the Scandinavian collaboration for Computer Assisted Radiation Therapy Treatment Planning (3); ACR-NEMA, a collaboration to establish a standard digital interface for medical imaging equipment (1); and the COST B2 Nuclear Medicine Project, a European collaboration in Nuclear Medicine. These three groups have essentially adopted the AAPM model for data storage. The data storage method proposed in this paper is similar to the AAPM model. It differs in that is is targeted directly to biomedical data, as reflected in the choice of keywords

[1]This work was supported by the U.S. Department of Energy at the Lawrence Livermore National Laboratory under contract W-7405-ENG-48 (P.D., L.M., and D.O.), by the Program for Analytical Cytology (University of California-LLNL) (D.S.), and by The Netherlands Project Team for Computer Science Research, SPIN (Project Three-Dimensional Image Analysis (J.M.).

[2]Address reprint requests to Phillip N. Dean, Biomedical Sciences Division, Lawrence Livermore National Laboratory, P.O. Box 5507 L-452, Livermore, CA 94551.

and data that are appropriate to store in the files. The AAPM model is targeted to medical data which are markedly different. The authors of this proposed standard are in communication with the authors of the AAPM Standard and will use common features in so far as is possible.

Image cytometry data can consist of many parts, with the sum referred to as a data set. The minimum set of data would contain the value of each pixel (picture element) in the image; e.g., in a 512 by 512 image there are 262,144 pixels. In a fluorescence image, these data will consist of the dye intensity measured at each pixel position. Usually, the value is contained in a single byte of 8 bits, for a maximum value of 255 units. A data set can also contain much more information: descriptions of the sample, experiment, and instrument; results of analysis of the data; a history of the processing of the data; etc. With the large variety of image cytometry systems in use today, the pixel data are written in various formats in binary files. In most cases, it would be impossible to even display the image without additional information, such as the number of bits per pixel, the order of pixel storage, the size of the image in x and y, the number of images in the file, etc. This paper proposes an image cytometry data file structure as a standard protocol to alleviate these problems and make a free interchange of data possible.

METHODS

There are two principal methods of storing image data. One method is to store all of the data relevant to a measurement in a single file. This is attractive from a data management point of view since one file would contain all information about an image. There are, however, several disadvantages: 1) A special program must be used to extract the descriptive information from the file to view, print, or modify it. 2) A fixed amount of space must be reserved for the descriptive information. 3) If the descriptive information is altered, the entire data file must be read and then written to a new file, which can require considerable time and at least temporarily consumes twice the storage space of the original file. This could be serious where a series of images are written to a single file of many megabytes. This problem could be alleviated if extra space (blank) were allocated for the text. Then only pointers to the information would have to be changed as additional text were entered. This is still very restrictive and potentially can waste a lot of disk space.

The second method of storing image data is to use several files. One file would contain the descriptive information, another file would contain the pixel information, and other files would contain results of analyzing the data, a history of the processing of the data, etc. There are several advantages of this method, such as: 1) The descriptive information can be written in the ASCII format, a directly printable, editable form. 2) The file containing the descriptive information is automatically dynamic in size; it can be of any length. 3) Results of image analysis procedures can be written into a separate file in a format that is compatible with word processors, spreadsheets, and data base programs on personal computers. The primary disadvantage of this method is that since the image data file contains only the pixel values, without the information that shows how the data are stored, the data file is virtually indecipherable. The solution to this problem is to store data by using a file structure where the raw data and its associated files are kept in a single directory. With modern methods of data backup and retrieval, loss of files should virtually never occur.

This proposed standard uses the second method of storage. It specifies that all of the data relevant to a measurement or series of related measurements, called here a data set, will be stored in a set of files. A data set will contain data from one or more images. Each image may contain a large number of objects. A data set requires a minimum of two files. One file, called the header file, contains information about the data set, such as the source of the images contained in the data set, their size, number of variables, etc. This file is written in ASCII and consists of a sequence of keywords and their values, organized in lines. Therefore, the file can be read and manipulated directly with editors. However, one must be careful in generating the file. If an editor is used to create the file, the result depends on the operating system used. For example, the VI editor in the UNIX operating system will automatically use the line feed character (octal 012) as a line terminator (upon pressing the "return" key). On the other hand, the EDT editor under the VMS operating system will place two characters at the beginning of a line, giving the number of characters in the line. It may also place a null character at the end of the line, to make the line have an even number of bytes. Both editors will correctly read and modify a file that has been created by using the line feed character at the end of each line. Both operating systems will also print the files correctly.

Some of the information contained in the header file is required; much of it is optional. The second file contains the image data, written in binary in the format specified within the header file. Additional files could contain the results of analysis of the data, descriptions of the experiment, and a history of the processing of the image(s). To provide an ability to share these data easily with other laboratories, these files will also be written in ASCII. A logical way to manage these files is to keep all files related to a given measurement in a single directory on the disk. If different users use a common data file to extract information in different ways, they could keep their analysis results files in their own directories with pointers to the raw data file.

In this standard we use the term *imel* (for image element) rather than *pixel* (for picture element) since the data can be n-dimensional. Thus imel can represent either two-dimensional pixels or three-dimensional

voxels (volume elements). An imel can have any number of dimensions.

DATA SET FILES

Header File

The header file is written in ASCII. It contains the formatting information that one must have in order to read the image data file. The first three lines of this file are fixed and mandatory. All three lines must be present and in the correct order. All other information is contained within four major categories: layout, representation, parameter, and history. The **layout** category is a required one. Without it, the data file could not be deciphered. The other three categories are not required since they are not strictly necessary for reading the data file. The user is, however, strongly advised to include them. The information is arranged as a sequence of lines, separated by a line separator character. Each line consists of a series of fields containing keyword(s) and value(s) combinations. Fields are separated with a field separator character. The first *line* in the file contains two bytes. The first byte contains the field separator character (e.g., the tab character, 011 in octal) and the second byte contains the line separator character (e.g., line feed, 012 in octal). Note: If the tab and line feed characters are used, the first line of this file may appear to be blank when the file is printed. The second *line* of the file contains the version number of the standard used in creating the file (e.g., ics_version 1.0). The third line of the file contains the name of the file being processed. The remainder of the information presented in this file is arranged into four major categories. These are:

Layout. This category is *required.* Included are the number of parameters (e.g., x, y, z, probe), the number of bits per image element (imel), the number of samples in each parameter (e.g., 256 in x), the order in which the parameter values are stored, the coordinate system used, and the number of significant bits per imel. The information contained in this category is sufficient to read and display the image data file.

Representation. Included in this category are the format of the image data, the type of data compression, if any, and the order of the bytes for multi-byte imels.

Parameter. Included in this category are the units and names of the parameters, the real-world coordinates of the images, scale factors used with the data, etc.

History. This category provides information about the experiment: the instrument used to obtain the data, the dyes used, etc. It also includes the names of other files associated with the data set, such as results of analysis.

Each of these major categories is divided into subcategories. A description of each subcategory is included in the full description of the standard. There are two minor categories, with no subcategories: **ics_version** and **filename.** The category **ics_version** contains the version of the image cytometry standard used to write the header file. The category **filename** contains the name of the image data set. This category is included so that when the header file is displayed or printed, the user will know which image the header belongs to. The name of the header file has an extension of "**.ics**", for Image Cytometry Standard (e.g., jan20f17.ics).

Data File

The name of the image data file has an extension of "**.ids**", indicating that it was written by using the Image Cytometry Data Standard, e.g., jan20f17.ids. This file contains the image data organized as specified in the header file. The data file is simply a list of the data (imel) values, stored in binary. In general, each imel value has a length in bits defined by the number of bits per imel (see the header file description). This allows the data word length to be defined dynamically, facilitating compatibility between machines with different data word lengths and/or allowing bit compression of the data. Data are stored in a continuous byte stream, with no delimiters. Figure 1 shows the layout of a three-dimensional data set. Note that the origin of each image is at the lower left corner. This would be described in the header file as cartesian coordinates.

Analysis File

An analysis file would typically contain information added to the data set after the original data were collected and stored. Examples are the number of objects in an image, their sizes, their integrated brightness, etc. While no specific structure for the ANALYSIS file is defined in this version of the ICS, the default format is ASCII. This allows both the header and analysis files to be edited by using standard editors available on all computers. The name of this file would have the extension "**.ias**" for Image Analysis Standard.

Comment File

A comment file could contain additional information about the image that is not included in the header file. For example, this file could contain the complete sample preparation protocol, a detailed description of the experiment, observations made during the experiment, remarks about the types of analysis, etc. Since the information contained within this file may be shared by a number of images within an experiment, and therefore a number of data files, there is no standard naming convention. The file is referenced within a heater file through the "history" category, as described in the standard.

Processing File

A processing file could contain information about the method, or sequence of commands, used to produce the image. For example, if TCL-Image (Perceptics, Knoxville, TN) were used to process the data, the list of commands used could be stored in a processing file. The name of the processing file would be stored in the header file by using the "**history**" category. This could

be especially useful when one wishes to process a new image in the same manner as a previous one was processed.

SUMMARY

This proposed standard, as presented in Appendix A, is the result of many discussions among several groups of investigators at the Lawrence Livermore National Laboratory and at the Delft University of Technology. It includes provisions for three-dimensional data and the inclusion of an unlimited amount of related data. It has evolved into a form that we fell is sufficiently flexible and still rigid where it needs to be, to be universally applicable. Since the text information, results of analyzing the data, etc., are in ASCII files, summaries of a numerical or other nature are easily extracted for transfer to other programs—e.g., databases, spreadsheets, and word processors, on other computers, e.g., Macintosh or IBM personal computers. For example, we routinely extract data from an analysis file and transfer it to a Macintosh computer over the Ethernet network for further analysis by using the Microsoft Excel program.

This paper presents our proposed standard. After a sufficient amount of time has passed to receive input from the research community at large, we will revise the standard, distribute it for further comment, and then publish it once again.

In its current form, the standard is used at the following laboratories:

Lawrence Livermore National Laboratory
University of California, San Francisco
Delft University of Technology, The Netherlands
University of Amsterdam, The Netherlands
State University of Utrecht, The Netherlands
State University of Leiden, The Netherlands

ACKNOWLEDGMENTS

The authors are indebted to many colleagues who have read the proposed standard and made valuable suggestions for its format.

LITERATURE CITED

1. ACR-NEMA Standards Publication NO. 300-1985: Digital Imaging and Communication. National Electrical Manufacturers Association (NEMA), Washington, DC, February, 1986.
2. Baxter BS, Hitchner LE, Maguire GQ Jr: A Standard Format for Digital Image Exchange. American Association of Physicists in Medicine (AAPM) Report No. 10. Am. Inst. of Phys. New York, NY, October, 1982.
3. Moeller T, Hyoedynmaa S: CART standard. CART Newslett No. 6:9–10, August 1987.
4. Rosenfeld A, Kak AC: Digital Picture Processing, Vol. 1. Academic Press, New York, 1982, pp 116–208.

APPENDIX A—ICS VERSION 1.0: A STANDARD METHOD OF STORING IMAGE DATA (19 JUNE 1989)

This Image Cytometry Standard for the storage of image data is presented item by item, with a full description and example for each entry. It also includes an example header file which illustrates the use of all categories and subcategories of information. There are two required files. A header file includes information necessary to be able to read the data file. It contains ASCII characters only. Some of the information in this file is mandatory and must be included. Other entries are optional, included to provide an opportunity for the user to keep relevant information about a measurement in one place and easily accessible. To denote the file as being written in the ICS format, its name has the extension "**.ics**". The second required file contains the image data. The name of this file will be the same as for the header file, with the extension "**.ids**". Other files containing processing information, comments, analysis results, etc., are optional. The only requirement is that they be written as ASCII files and referenced in the header file.

THE HEADER FILE

The following paragraphs describe the entries in a typical header file, and provide some examples. The descriptions are arranged by category and then subcategory. Note that all category and subcategory names are in lower case. The first three lines of the file are mandatory and must be in the order shown. All other lines can be in any order. However, the order shown in Table 1 is suggested for ease in reading and interpreting the information. Also required are all entries described for the **layout** category. Entries for all other categories are optional, although the user will find it very useful to include input for the **representation** and **parameters** categories. For many of these entries, there are default values which will be given when the entry is explained. In this discussion, the term imel (image element) is used rather than the term pixel (picture element), since the data can be n-dimensional. Thus imel can represent either two-dimensional pixels or three-dimensional voxels (volume elements).

Except for the first line, the information in this file is contained in keyword/value combinations. The file consists of lines of data, separated by a line separator character. Each line consists of a number of fields, separated by a field separator character. Keywords are placed into fields and consist of the category and subcategory names. All values for a category/subcategory are also placed in fields in the same line. In the following descriptions, the examples come from Table 1.

LINE 1

The first line contains the 8-bit ASCII characters for the field and line separators. In principle, any two characters could be used for defining the separators, as long as these characters are unique and not used within any keyword or value.

<HT><LF>

The horizontal tab character (octal 011) is suggested as the field separator and the line feed character (octal

012) as the line separator character. If these characters are used and the header file printed, these characters would not be visible. In this document we will use these characters.

LINE 2

This line contains the version number of the ics standard used to write the data file.

ics_version 1.0

This is the first version of the ics standard (**1.0**). The first field of the line contains **ics_version** followed by a <**HT**> character and then **1.0**. There is a <**LF**> character at the end of the line.

LINE 3

filename jan20f17

The category **filename** contains the name of the image data set. This category is included so that when the header file is displayed or printed, the user will know what image the file refers to. The extension to the file name would be "**.ics**" for the header file; the complete file name would be jan20f17.ics.

Major Categories and Subcategories

The remaining lines of the header file contain information about the image, organized by category and subcategory. The following paragraphs describe the entries, give the default values if they exist, and present examples. The examples come from the complete header file shown in Table 1. These entries can be placed in the file in any order, although the order presented here is suggested for clarity.

1. Category layout

This category defines the layout of the image data file using five subcategories, as shown below. All subcategories of the **layout** category are required entries in the header file.

A. Subcategory parameters

The subcategory **parameters** specifies the number of parameters stored in the data file. This entry consists of three fields separated by two horizontal tab characters, with a line feed character at the end of the line.

layout parameters 5

This example shows that five parameters are included in the file. The first parameter, as described below, is used to specify the number of bits used to store the value of an imel. The remaining parameters correspond to the dimensions of the image data. Thus this example shows that a four-dimensional data set was stored in the file.

B. Subcategory order

The subcategory **order** describes the order in which the data are written. There are five reserved keywords for **order.** They are:

bits The number of bits used to store the value of an imel in the image data file. This is a required entry and must be placed immediately after the **order** keyword.

x,y,z The order in which the data are written. If the order were **x y,** the data would be written one row at a time. If **y x,** the data would be written by one column at a time. If **z** is present, the data consist of a set of images (e.g., a set of serial sections).

probe The presence of this keyword shows that data from more than one probe (dye) is contained in the file. Its values are the names of the probes.

Additional keywords could be used to describe other parameters, e.g., time, etc. In the example of Table 1, the number of parameters is five. Therefore, there are five fields following the keyword **order.**

layout order bits x y z probe

The order specified by this line is **x**-coordinate first and **y**-coordinate second. The "**z**" indicates that this is a sequence of images, stored frame by frame, and "**probe**" indicates that data were collected by using more than one fluorescent dye. The x,y,z sequence is repeated for each fluorescent dye.

C. Subcategory sizes

The subcategory sizes specifies the size (range) of each parameter listed under the subcategory **order**. The values are entered in the same order as the keywords. The entry for the keyword **bits** contains the number of bits per imel.

layout sizes 8 64 64 32 3

In this example, there are 8 bits per image element. There are 32 images of 64 samples in both x and y. This constitutes a three-dimensional image, e.g., a serial-sectioned cell. Measurements were made for three probes yielding three 3-dimensional images. Therefore the data file contains a total of 96 individual xy images.

D. Subcategory coordinates

The subcategory **coordinates** defines the coordinate system used to store the data. The most common systems are cartesian and video. There are, therefore, two reserved keywords for this subcategory:

video In a video coordinate system, the origin is in the upper left corner of the image. X increases to the right and Y increases downward.

cartesian In the cartesian coordinate system, the origin is at the lower left corner of the image. X increases to the right and Y increases upward.

Table 1
An Example of a Complete Header File

<HT><LF>		
ics_version	1.0	
filename	jan20f17	
layout	parameters	5
layout	order	bits x y z probe
layout	sizes	8 64 64 32 3
layout	coordinates	video
layout	significant_bits	8
representation	format	integer
representation	sign	unsigned
representation	compression	uncompressed
representation	byte_order	1
parameter	origin	0 10.2 3.2 −6.3 0
parameter	scale	1.0 0.004 0.004 0.008 0
parameter	labels	intensity x-position y-position z-position probe
parameter	units	relative mm mm mm undefined
parameter	probe	DAPI FITC TR
history	date	22APR89
history	time	10:23:12.45
history	title	Optical sectioning of human lymphocyte nuclei
history	computer	VS3200
history	system	ULTRIX32
history	laboratory	LLNL
history	operator	F. Bighelp
history	researcher	J. Scientist
history	sample	peripheral blood
history	source	M. Jones
history	microscope	ZEISS Axiophot
history	objective	100× 1.3 NA
history	instrument	MRC500
history	camera	Photometrics TC215
history	software	TCL-Image
history	DAPI	excitation 364 emission 481 wb
history	DAPI	PMT 456V
history	DAPI	target nucleus
history	FITC	excitation 514 emission 530 bp
history	FITC	PMT 1250V
history	FITC	target chromosome 9
history	TR	excitation 514 emission 680 wb
history	TR	PMT 300V
history	TR	target chromosome 1
history	comments	This is the first of a set of 5 experiments.
history	comment_file	jan20f17.cmt
history	analysis_file	jan20f17.ias
history	related_files	jan20f17.txt jan21f14.ias
history	processing_file	raf.tip

Other coordinate systems could be polar, cylindrical, spherical, etc. Specific keywords for these systems are not defined in this version of the standard.

layout coordinates video

This image was stored in the video coordinate system.

E. Subcategory significant_bits

In some cases, the number of bits per imel, as specified in the **layout sizes** category, does not reflect the number of significant bits per imel. For example, if one acquires an image with a camera that has 12-bit resolution, the data could be stored in a 16-bit word (two 8-bit bytes). The data must be in the low order bits of the word, which means that the upper 4 of the most significant bits would not be used and must be set to zero (0). In this case, the **layout bits** category would have a value of 16 while the **significant_bits** keyword would have a value of 12.

layout significant_bits 8

The number of significant bits per imel is 8. In the example of Table 1, this is the same as the number of bits per imel, i.e., 1 byte.

2. Category representation

This category specifies how the image data are to be interpreted. Included under this category is the imel format (i.e., integer, real or complex), the order of bytes in multi-byte imels, and whether the imel value is signed or unsigned, or compressed.

A. Subcategory byte_order

The subcategory byte-order specifies the order of the bytes in a multi-byte imel. With **1** representing the least significant byte and **4** the most significant byte, data are interpreted in the following way:

representation byte_order 1 2 3 4 (32-bit imels)

representation byte_order 1 2 (16-bit imels)

representation byte_order 1 (8-bit imels)

In all of these examples, the least significant word is stored (and read back) first. The order in which the bytes appear in the specification is the order in which they appear in the data file. For example, with the order specified as "**1 2 3 4**", in reading the file, byte **1** (least significant byte) of the data word would be read first, byte **2** second, byte **3** third, and byte **4** (most significant byte) would be read last. If the order were "**2 1 4 3**", byte **2** would be read first, byte **1** second, byte **3** third, and byte **4** would be read last. If there is only 1 byte per imel, this line is not necessary. For multi-byte imels, byte order is very important and inclusion of this line is mandatory.

representation byte_order 1

Since there is only 1 byte (8 bits) per imel, this line does not add any information and could be deleted.

B. Subcategory format

This subcategory specifies the format of the imels in the image data file. Three types of values are supported by this version of the ICS Standard:

integer The imel values are stored as integers. The size of the integer is specified by the number of bits per imel (see **layout sizes**).

real The imel values are stored as real values. In this version of the ICS Standard, real is defined to comply with the IEEE floating-point standard.

complex The imel values are stored as complex numbers. For each imel, the real part is stored first and the imaginary part stored last. Each part is stored as a real value, as described above. The number of bits per imel is the total number of bits in both the real and imaginary parts.

If this line is not present in the header, the format defaults to integer.

representation format integer

This line shows that the format of the imel values is integer. Since this is the default, this line could be deleted.

C. Subcategory sign

The subcategory **sign** specifies whether the imel values are written as signed or unsigned numbers. This subcategory is used only where the format is integer. Two options are available in this version of the ICS Standard:

signed The imel values are signed integers, i.e., two's complement representation is used.

unsigned The imel values are unsigned integers.

If this line is not present, the default value is **unsigned** for integers and **signed** for real and complex formats.

representation sign unsigned

This example shows that the imel values are unsigned integers. Since this is the default value, this line could be deleted.

D. Subcategory compression

Image data are sometimes stored in a compressed form to conserve disk space. This can be an important consideration when storing several three-dimensional image data sets. The subcategory **compression** specifies whether the data are written in a compressed or uncompressed form. There are many methods of compressing image data, e.g., Karhunen-Loeve (KL), Fourier, Hadamard, cosine, predictive, and Huffman. See Rosenfeld and Kak (4) for a discussion of these methods. If one of the methods of data compression listed above or a different one is used in storing the data, the value of the keyword would be the name of the method. The default value is **uncompressed.**

representation compression uncompressed

In this example, the data are not compressed. Since this is the default, this line is not required.

3. Category parameter

The **parameter** category specifies how the image data relate to real-world values. Included in this category are the real-world origin, scale, units, and labels. This calibration information can be vital to the quantitative analysis of the image data.

A. Subcategory origin

The subcategory **origin** is included to provide an offset for data where an image is not taken beginning at an origin of (0,0,0) in world coordinates. The values give the offset in x, y, and z. For example, if an image is taken at some position relative to the corner of a microscope slide, then the origin values for x and y would be the distance from the corner of the slide to the beginning of the image. The origin in z could be the depth below the surface of the coverslip. Values in the fields following the keyword **origin** correspond in order to those in the category **layout order.**

parameter origin 0 10.2 3.2 −0.0063 0

In this example, the first value, **0,** indicates that an imel value of zero corresponds to a real word value of zero; i.e., there is no offset in the measurement of the imel values. The next three values correspond to the real-world location of the first imel in the image data file. The units are the same as for the x, y, and z pa-

rameters, as described later in the **units** subcategory. This entry shows that the image was obtained at a location 10.2 mm from the corner of the slide in the x direction and 3.2 mm in the y direction, and 0.0063 mm (6.3 μm) from the bottom of the coverslip. The last value is zero since the fifth entry under **layout order** has no units (number of probes).

B. Subcategory scale

The subcategory **scale** specifies the separation between successive imels, in the units presented in the subcategory **units.** These entries also correspond to those in the **layout order** category. The scale value corresponding to bits specifies how a change of one unit in the imel value (e.g., from 35 to 36) relates to change in real-world units (e.g., watts per square centimeter). The scale values for x,y, and z relate the distance between adjacent imels to a real word distance (e.g., imels per micron).

parameter scale 1.0 0.004 0.004 0.008 0

The first value after scale corresponds to bits in the layout order category. This indicates that a change of 1 in an imel value corresponds to a real-world change of 1. However, this relationship is relative, i.e., not calibrated, as indicated by the subcategory units. Although the imel values are not calibrated, they are linearly related to the real world. Otherwise the value would be zero and a lookup table would be provided. The next three field values correspond to the **x, y,** and **z** scale values, respectively. Referring to the subcategory units, the x and y imels are separated by 0.004 mm and the z imels are separated by 0.008 mm. The last value (0) corresponds to the parameter **probe.** The value of zero means that there is no linear scale for this parameter. This also means that further information about this parameter will be given under the subcategory **probe.**

This discussion has dealt only with linear mappings from image values and coordinates to real-world values and coordinates. As mentioned above, a value of zero means that the scale for this parameter is nonlinear. Nonlinear, nonuniform, or other mappings require the use of a lookup table. The lookup tables are provided through subcategories of the **parameter** category, one for each nonlinear mapping. The subcategories have the name of the parameter as listed in the layout order category, e.g., **x, y,** and **z.** The number of entries in the lookup table is given by the value specified in the **layout size** category. For example, for a nonlinear mapping of the **x** parameter data in Table 1, the line would be **parameter x** followed by the 64 values of the lookup table.

C. Subcategory labels

This subcategory provides labels for each of the parameters.

parameter labels intensity x-position y-position z-position probe

D. Subcategory units

This subcategory specifies the units of the parameters. The keywords can be any appropriate measurement unit or one of two keywords:

relative This keyword indicates that the corresponding parameter was not calibrated and that the values are relative.

undefined The units are undefined for the corresponding parameter.

parameter units relative mm mm mm undefined

These values indicate that the imel values are relative and the x, y, and z coordinates are measured in mm. The probe measurement unit is undefined.

E. Subcategory probe

This subcategory specifies the use of more than one probe (dye) in the measurement. Its values are the names of the probes. This is a required entry if the value of **probe** is given as zero (0) in the **parameter scale** category (see description above).

parameter probe DAPI FITC TR

This example shows that the three probes used were the fluorescent dyes, DAPI, FITC, and TR.

4. Category history

The **history** category is intended to include information that might provide a better description of an experiment. The entries could be decoded by the user and used in reports, for computations (e.g., elapsed times), etc.

history	date	22APR89
history	time	10:23:12.45 hr:min:sec
history	title	Optical sectioning of nuclei
history	computer	VS3200
history	system	ULTRIX 32
history	laboratory	LLNL
history	operator	F. Bighelp
history	researcher	J. Scientist
history	sample	peripheral blood
history	source	M. Jones
history	microscope	ZEISS Axiophot
history	objective	100× 1.3 NA
history	instrument	MRC500
history	camera	Photometrics TC215
history	software	TCL-Image

The following nine lines illustrate how the **history** category is used to further explain other entires. The **layout order** entry specified that **probe** was one of the parameters. The **parameter probe** entry gave the

names of the probes. The **history** entries specify how the parameters were measured.

history	DAPI	excitation	364	emission	418WB
history	DAPI	PMT	456V		
history	DAPI	target	nucleus		
history	FITC	excitation	488	emission	530BP
history	FITC	PMT	1250V		
history	FITC	target	chromosome 9		
history	TR	excitation	514	emission	680WB
history	TR	PMT	300V		
history	TR	target	chromosome 1		

A. Subcategory comments

The subcategory comments allows for the use of single-line comments.

history comments Hybridization to chromosomes 1, 3 & 9.

This entry could be used to include random comments, restricted to one line.

B. Subcategory analysis_file

The subcategory **analysis_file** allows for the inclusion of the name of a file that contains the results from analysis of the image data. The extension to the file name should be "**.ias**".

history analysis_file jan20f17.ias

C. Subcategory comment_file

The subcategory **comment_file** allows for the inclusion of the name of a file that includes extensive comments, more than can be entered by using the **comments** subcategory. The extension for the file name is "**.cmt**".

history comment_file jan20f17.cmt

D. Subcategory related_files

The subcategory **related_files** allows for the inclusion of names of other files that contain information related to the experiment. These could be calibration images, text files, etc.

history related_files jan20f17.txt

E. Subcategory processing_file

The subcategory **processing_file** allows for the inclusion of the name of a file that includes the commands used to process the image data. This makes it possible to process other images with the same set of commands, yielding comparable results.

history processing_file raf.tip

This example includes the name of a command file generated with the TCL-Image program (Perceptics, Knoxville, TN).

Entires under the **history** category can be anything the user desires. It is suggested that the format described above be used so that the data can be easily decoded at other laboratories. We also suggest that the subcategories shown be used. Additional ones can be defined by the user. To distinguish them from ICS Standard subcategories, user-generated subcategories should be named in UPPER CASE.

This data file standard is in use at the Lawrence Livermore National Laboratory, at the Delft University of Technology, and at several other laboratories. For other laboratories willing to use the standard, we will provide two subroutines that are necessary to implement the standard:

1. A subroutine to read the HEADER file, extract the pertinent information, and read the DATA file.
2. A subroutine to create the DATA file and write a set of data to it, and create and write the appropriate information to a HEADER file.
3. We will also provide a pair of files (.ics and .ids) that can be used to test a laboratory's implementation of the subroutines.

This proposed standard is still subject to revision. We invite comments from other investigators to help us make the standard responsive to the research community.