

# Appendix E: PIF – Parametric Image Format

InnovMetric Software has devised PIF, a binary file format for describing 3D parametric images. This appendix introduces the concept of parametric images and gives a complete description of the PIF format. Note that PIF files must be written using a big-endian byte-ordering scheme.

## E.1 Fundamentals

In this document, a parametric image is defined as a 3D surface mesh that can be mapped and interpolated onto a 2D parametric space. The PIF format supports the definition of planar and cylindrical 2D parametric spaces. It should be noted, however, that PolyWorks does not support cylindrical images.

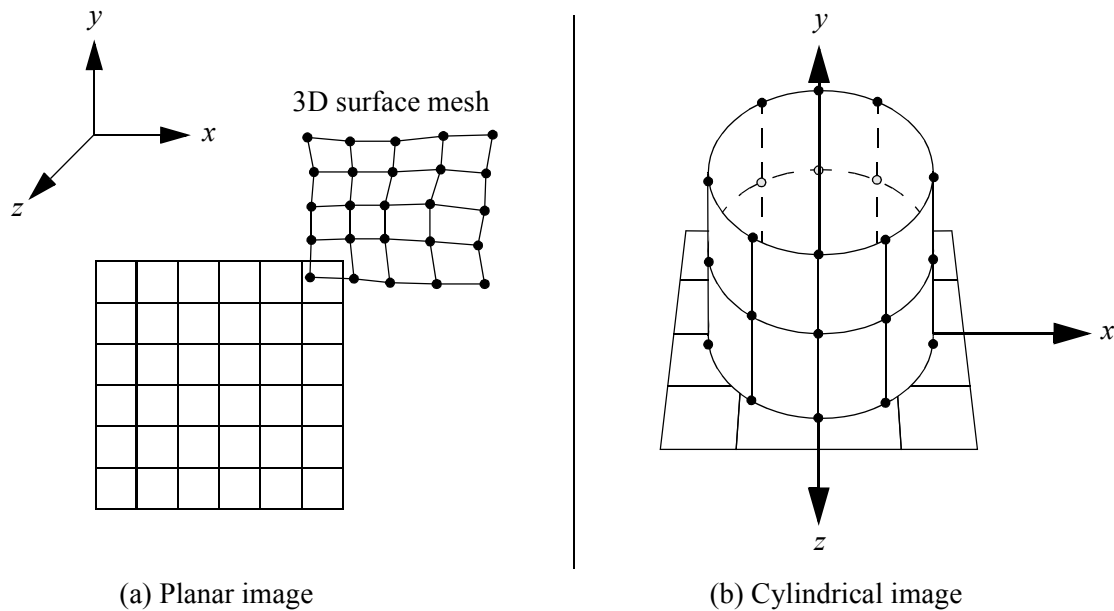
### E.1.1 Planar and cylindrical parametric images

A planar image consists of a 3D surface mesh that can be mapped onto the 2D parametric space of a plane. A cylindrical image consists of a 3D surface mesh that can be mapped onto the 2D parametric space of a cylinder. Examples of planar and cylindrical images are shown in [Figure E.1](#).

### E.1.2 Raw and interpolated parametric images

In this document, a raw parametric image is defined as a 3D surface mesh measured by a 3D imaging device. A raw parametric image is described by a set of  $(x, y, z)$  point coordinates and the connectivity information between these points. There are two ways to specify a raw parametric image in the PIF format. A 3D surface mesh can be described by an array of data points or by a polygonal mesh. As shown in [Figure E.2 \(a\)](#) and [\(b\)](#), the data points in a raw parametric image need not be distributed in any particular way in the 2D parameterization space.

For several algorithms used in the PolyWorks modules, however, it is more convenient to describe a 3D surface mesh by an array of data points distributed on a square grid in the 2D parameterization space (see [Figure E.2 \(c\)](#)). In this document, a parametric image parameterized on a square grid is called an interpolated parametric image. PolyWorks provides interpolation routines to transform raw parametric images into interpolated parametric images.




---

*Figure E.1 Examples of parametric images. In (a), the 3D surface mesh can be mapped onto the 2D parametric space of a plane. In (b), the 3D surface mesh can be mapped onto the 2D parametric space of a cylinder.*

---

## E.2 Coordinate systems in the PIF format

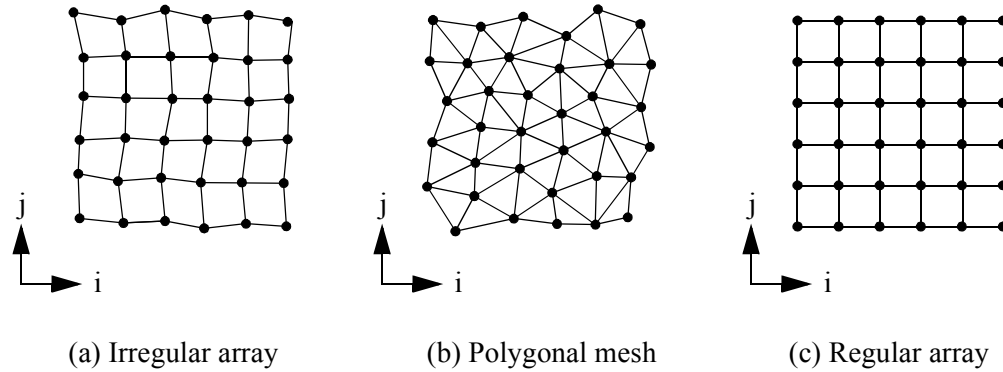
Three coordinate systems are used in the PIF format, as depicted in [Figure E.3](#). This section describes the role of these coordinate systems.

### E.2.1 Data Coordinate System (DCS)

The Data Coordinate System (DCS) is the standard Cartesian coordinate system in which the  $(x, y, z)$  point coordinates of a raw parametric image are expressed. When several 3D images of a single object are acquired, each image has its own individual DCS. An alignment step is thus needed to transform these individual DCSs into a common coordinate system. As shown in [Figure E.4](#), PolyWorks alignment matrices perform a rigid transformation on the image Data Coordinate Systems. These alignment matrices are saved in the image information files of a group directory (see [Appendix C](#)).

### E.2.2 Intermediate Coordinate System (ICS)

An Intermediate Coordinate System (ICS) has been defined in the PIF format. The ICS is linked to the DCS by a user-defined rigid transformation  $T$ . A 3D image is transformed




---

*Figure E.2 Two raw parametric images and one interpolated image mapped onto a 2D parametric space  $(i, j)$ . In (a), the mesh is described by an array of data points that may be irregularly distributed. In (b), the mesh is described by a triangulation. In (c), the mesh is described by an array of data points distributed on a square grid.*

---

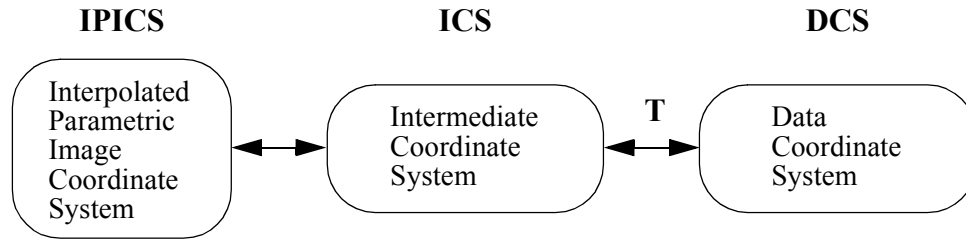
into the ICS prior to its mapping on a planar or cylindrical surface. Some information about the position and orientation of the parameterization surfaces is extracted from the ICS. If the parametric image is planar, the  $z$  axis of the ICS is considered to be the plane normal vector. If the parametric image is cylindrical, the  $y$  axis of the ICS forms the axis of the cylinder.

### E.2.3 Interpolated Parametric Image Coordinate System (IPICS)

The last coordinate system in the PIF format is the Interpolated Parametric Image Coordinate System (IPICS). The IPICS is a scaled 2D parametric coordinate system defined on a planar or cylindrical surface and is used to represent interpolated parametric images.

Points expressed in a Cartesian 3D space are transformed into a 2D parametric coordinate system by means of an operation called parameterization. The parameterization operation consists in mapping a point  $(x, y, z)$  onto a 2D parametric space to find its  $(i, j)$  coordinates, and then computing the value of a function  $f(i, j)$  which can be used to retrieve the original point  $(x, y, z)$ . The planar and cylindrical parameterizations that are performed in the PIF format are depicted in [Figure E.5](#). If the parametric image is planar, the  $i$  and  $j$  coordinates are respectively equal to the  $x$  and  $y$  coordinates of a point, and function  $f(i, j)$  is equal to the  $z$  coordinate. If the parametric image is cylindrical instead, then  $i$  is equal to the smallest positive angle between the  $(x, 0, z)$  vector and the  $z$  axis, and  $j$  is equal to  $y$ . In the cylindrical case, the  $f(i, j)$  function is equal to the length of the  $(x, 0, z)$  vector.

The main purpose of the IPICS is to allow the representation of interpolated parametric images. An interpolated parametric image can be efficiently represented by a rectangular array of floating-point numbers providing values of function  $f$ . The interpolation



*Figure E.3 Three coordinate systems are used in the PIF format. The measured 3D points in a raw parametric image are given in the Data Coordinate System (DCS). A rigid transformation  $T$  is then performed on the raw image prior to its mapping on a planar or cylindrical surface, bringing the 3D image into the Intermediate Coordinate System (ICS). Finally, the 3D image is parameterized and interpolated by PolyWorks. The resulting interpolated parametric image is defined in the Interpolated Parametric Image Coordinate System (IPICS).*

procedure is performed once the original 3D surface mesh has been parameterized, and the  $(i, j)$  values have been scaled to ensure that the parametric distance between two consecutive horizontal or vertical neighbors in the interpolated square grid is equal to 1. The origin of the IPICS is the lower left corner of the array.

### E.3 Specifying a raw parametric image in the PIF format

As depicted in [Figure E.2](#), the PIF format offers two ways to specify a raw parametric image. You may either specify an array of  $(x, y, z)$  point coordinates or a polygonal mesh. It should be noted that in a cylindrical array of point coordinates, the array wraps around the cylinder axis such that the rightmost column is connected to the leftmost column. The point coordinates in a raw parametric image are given in the Data Coordinate System. The sequence of operations performed by PolyWorks on a raw parametric image is shown in [Figure E.6](#).

A raw parametric image, defined in the DCS, is first transformed into the ICS by means of a 4x4 rigid transformation matrix. Let us assume matrix  $M_{DI}$  having the following form:

$$M_{DI} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \\ m_{13} & m_{14} & m_{15} & m_{16} \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{D-1})$$

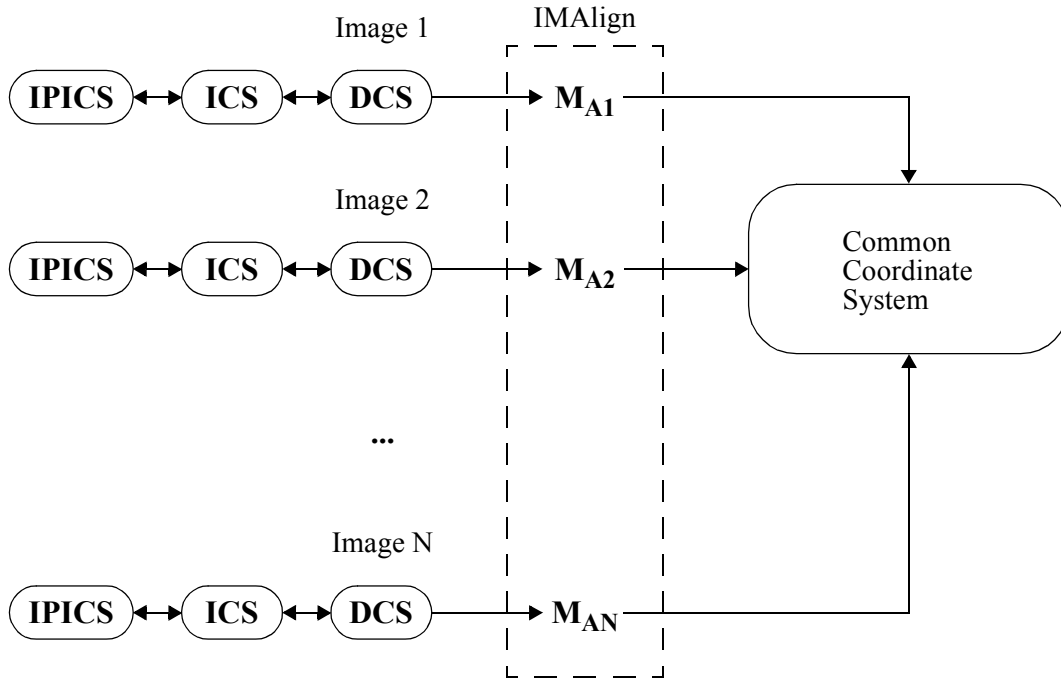


Figure E.4 This figure shows the relationship between the PIF coordinate systems and the image alignment matrices for a group of  $N$  images. The  $N$  image alignment matrices  $M_{A1}$ ,  $M_{A2}$ , ...,  $M_{AN}$  perform rigid transformations on the 3D images defined in their individual DCS in order to define a Common Coordinate System. These image alignment matrices are saved in the image information files of a group directory (see [Appendix C](#)).

A point given in the DCS is then transformed into the ICS by performing the following operation:

$$\begin{bmatrix} x_{ICS} \\ y_{ICS} \\ z_{ICS} \\ 1 \end{bmatrix} = M_{DI} \begin{bmatrix} x_{DCS} \\ y_{DCS} \\ z_{DCS} \\ 1 \end{bmatrix} \quad (D-2)$$

The parameterization operation is performed on raw images after their transformation into the ICS. Consequently, the PIF format offers the flexibility of specifying a raw image in virtually any coordinate system, as long as matrix  $M_{DI}$  transforms the measured surface mesh into an appropriate coordinate system for parameterization. If the parametric image is planar, the parameterization is performed as follows:

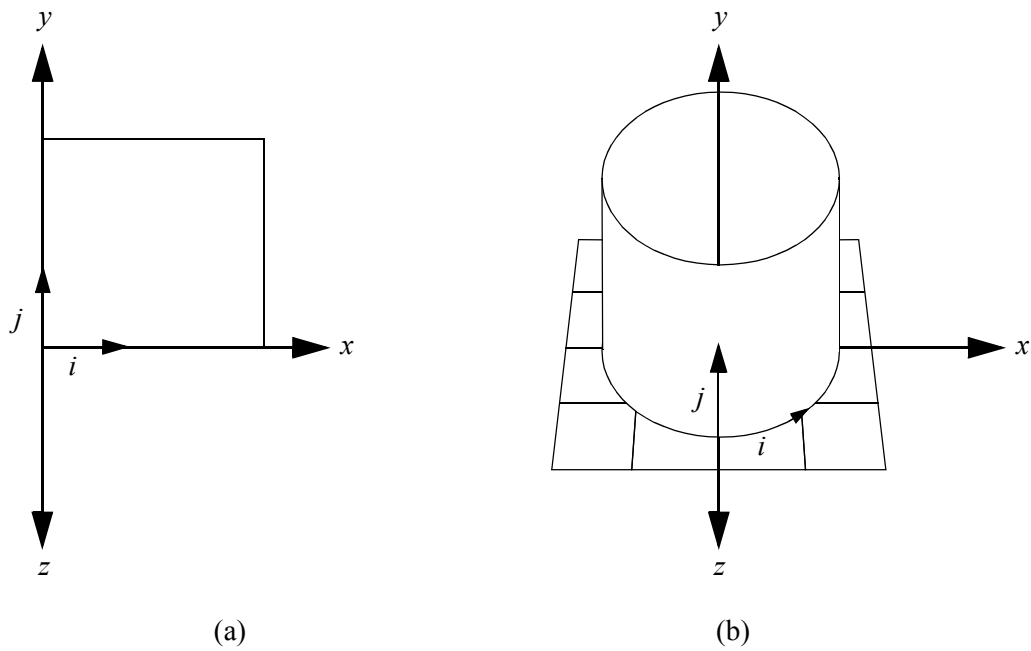


Figure E.5 Planar (a) and cylindrical (b) parameterizations in the PIF format.

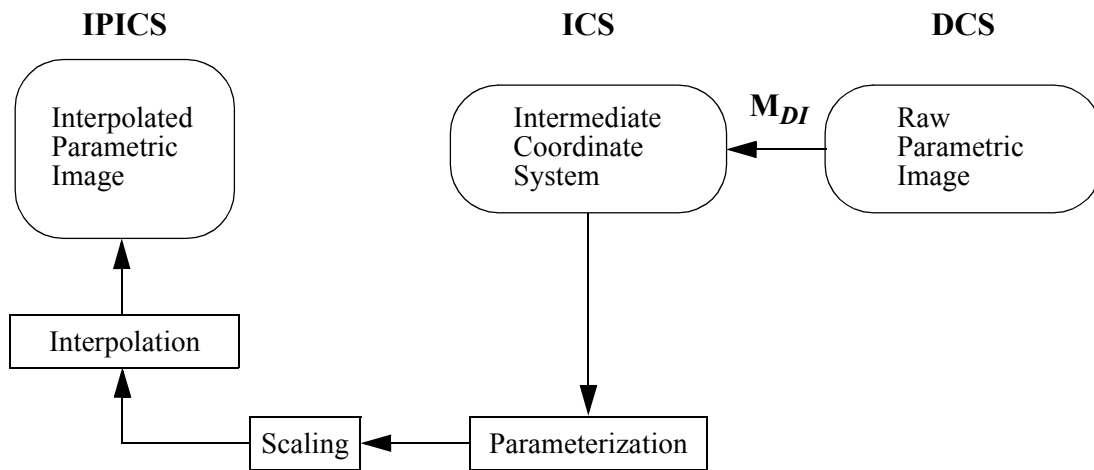


Figure E.6 Sequence of operations applied to a raw parametric image to generate an interpolated parametric image.

$$\begin{aligned}
 i &= x \\
 j &= y \\
 f(i, j) &= z
 \end{aligned}
 \tag{D-3}$$

If the parametric image is cylindrical, the parameterization is performed as follows:

$$\begin{aligned} i &= \operatorname{atan}\left(\frac{x}{z}\right) \\ j &= y \\ f(i, j) &= \sqrt{x^2 + z^2} \end{aligned} \tag{D-4}$$

The  $(i, j)$  parametric coordinates are then divided by two scaling factors  $i\_scale$  and  $j\_scale$ :

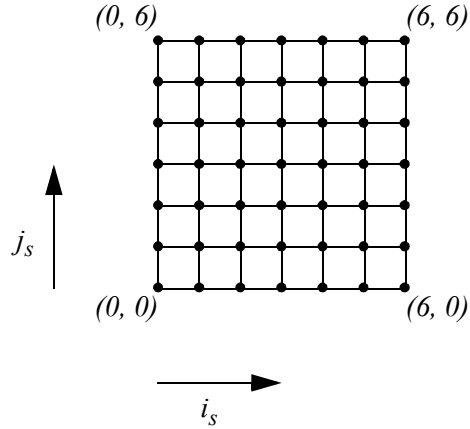
$$\begin{aligned} i_s &= i/i\_scale \\ j_s &= j/j\_scale \end{aligned} \tag{D-5}$$

The raw image can then be interpolated by PolyWorks. Given the scaling factors, PolyWorks automatically determines the origin of the IPICS that minimizes the size of the interpolated image. You do not have direct access to the IPICS when the interpolation is performed by PolyWorks.

## E.4 Specifying an interpolated parametric image in the PIF format

It is possible to directly specify an interpolated parametric image in the PIF format. An interpolated image is represented by a 2D array of floating-point numbers providing the values of function  $f$  for a square grid of integer  $(i, j)$  coordinates, as shown in [Figure E.7](#). In the IPICS, the parametric distance between two horizontal or vertical neighbors in the grid is equal to 1. The origin of the coordinate system is the lower left corner of the array. It should be noted that in a cylindrical interpolated image, the array wraps around the cylinder axis such that the rightmost column is connected to the leftmost column. PolyWorks aligns 3D parametric images with respect to their DCS. Therefore, the PIF format requires that the sequence of operations transforming the IPICS into the DCS be completely specified. This sequence of operations is depicted in [Figure E.8](#).

Given an interpolated array of floating-point numbers, the parametric coordinates  $i_s$  and  $j_s$  of a particular array element are directly obtained from its position in the array (see [Figure E.7](#)). The  $(i, j)$  coordinates are then obtained by multiplying the  $(i_s, j_s)$  coordinates by the two scaling factors:




---

*Figure E.7 In the IPICS, an interpolated 3D image is represented by a 2D array of floating-point numbers distributed on a regular square grid. The parametric distance between two consecutive horizontal or vertical neighbors is equal to 1. The origin of the IPICS is the lower left corner of the array.*

---

$$\begin{aligned} i &= i_s \cdot i\_scale \\ j &= j_s \cdot j\_scale \end{aligned} \tag{D-6}$$

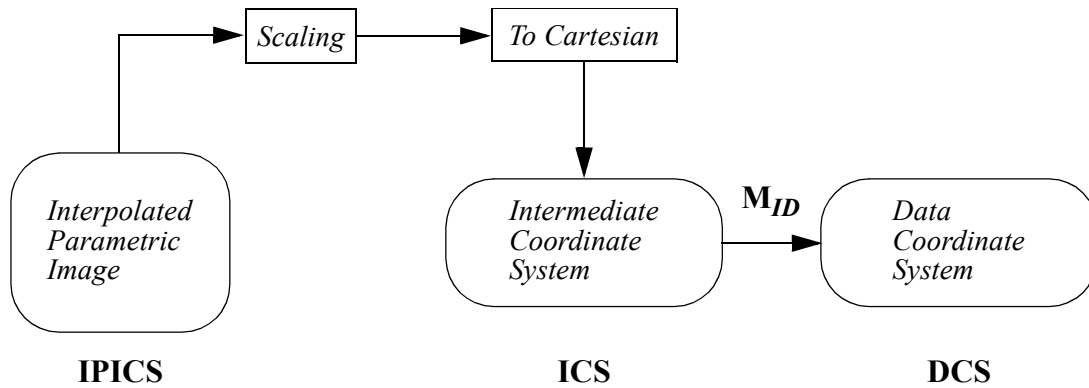
If the parametric image is planar, the  $(x, y, z)$  coordinates in the ICS are obtained as follows:

$$\begin{aligned} x &= i \\ y &= j \\ z &= f(i, j) \end{aligned} \tag{D-7}$$

If the parametric image is cylindrical, the  $(x, y, z)$  coordinates in the ICS are obtained as follows:

$$\begin{aligned} x &= f(i, j) \cdot \sin i \\ y &= j \\ z &= f(i, j) \cdot \cos i \end{aligned} \tag{D-8}$$

A point defined in the ICS is then transformed into the DCS by means of the rigid transformation matrix  $M_{ID}$ :




---

Figure E.8 Sequence of operations transforming an interpolated parametric image into the Data Coordinate System.

---

$$\begin{bmatrix} x_{DCS} \\ y_{DCS} \\ z_{DCS} \\ 1 \end{bmatrix} = M_{ID} \begin{bmatrix} x_{ICS} \\ y_{ICS} \\ z_{ICS} \\ 1 \end{bmatrix} \quad (D-9)$$

## E.5 Specification of the PIF format

PIF format files are identified by the “.pif” extension. A PIF file may comprise up to three blocks of information. The first block is a header that contains useful information about the file structure. The second block of information specifies the image 3D data. The optional third block of a PIF file contains color information. The rest of this section explains the structure of each block of information in a PIF file. This specification assumes that the data type **long** is 4 bytes long.

### E.5.1 Block 1: Header

The header part of a PIF file is 512 bytes long and contains useful information for reading the rest of the file. The following structure may be used to read the header of a PIF file (C language syntax):

```

struct header
{
    char    format_version[64];
    char    user_comments[128];
    char    dummy1[8];

```

```

long    image_param_flag;
long    image_data_type;
float   invalid_point;
long    array_width;
long    array_height;
long    data_block_length;

long    scale_flag;
float   i_scale;
float   j_scale;
long    transfo_matrix_flag;
double  transfo_matrix[16];

long    image_color_flag;
long    color_block_length;

long    camera_position_flag;
float   camera_x;
float   camera_y;
float   camera_z;

long    dummy2 [30];
};

```

**format\_version:** A 64-byte field reserved for an identifier string which gives the name of the format and the version number. InnovMetric currently writes “PIF Format v2.0” in this field.

**user\_comments:** A 128-byte field where you may write any information about the model.

**dummy1:** An 8-byte field that may be used for special characters.

**image\_param\_flag:** A flag indicating the type of image parameterization.

If 0, the parameterization is PLANAR.

If 1, the parameterization is CYLINDRICAL.

**image\_data\_type:** A flag indicating the type of data written in the 3D data block:

0 – the data block contains a 2D array of floating-point numbers expressed in the IPICS. This data type describes a parametric image that is already interpolated.

1 – the data block contains a 2D array of  $(x, y, z)$  coordinates expressed in the DCS. This data type describes a raw

	<p>parametric image. Connectivity between 3D points is derived from the connectivity in the array.</p> <p>2 – the data block contains two strings specifying a) the path to an external polygonal file, and b) the corresponding polygonal file format.</p>
<code>invalid_point:</code>	<p>If <code>image_data_type</code> is 0, an array element whose value is equal to <code>invalid_point</code> should be considered as an invalid point.</p> <p>If <code>image_data_type</code> is 1, a point whose <i>z</i> coordinate is equal to <code>invalid_point</code> should be considered as an invalid point.</p>
<code>array_width:</code>	<p>Width of the 2D array (not used if the data is contained in an external polygonal file).</p>
<code>array_height:</code>	<p>Height of the 2D array (not used if the data is contained in an external polygonal file).</p>
<code>data_block_length:</code>	<p>Length in bytes of the data block.</p> <p>If <code>image_data_type</code> is 0 or 1, the data block length is equal to the size of the 2D array in bytes.</p> <p>If <code>image_data_type</code> is 2, <code>data_block_length</code> should be equal to 1024 bytes.</p>
<code>scale_flag:</code>	<p>A flag related to the scaling factors:</p> <p>0 – the scaling factors should not be used.</p> <p>1 – the scaling factors should be used. If <code>image_data_type</code> is 0, the scaling factors must be specified. If <code>image_data_type</code> is 1 or 2, the scaling factors are optional.</p>
<code>i_scale:</code>	<p>Scaling factor for the <i>i</i> coordinate. If the parameterization is PLANAR, <code>i_scale</code> is specified in image units. If the parameterization is CYLINDRICAL, <code>i_scale</code> is specified in degrees.</p>
<code>j_scale:</code>	<p>Scaling factor for the <i>j</i> coordinate, specified in image units. If the parameterization is PLANAR, <code>i_scale</code> and <code>j_scale</code> should be equal.</p>

- `transfo_matrix_flag`: A flag indicating the transformation performed between the DCS and the ICS:
- 0 – the transformation matrix between the DCS and the ICS is an identity matrix. In this case, the DCS and the ICS are equivalent and the data contained in field `transfo_matrix` is not used.
  - 1 – the field `transfo_matrix` specifies a matrix  $M_{DI}$  that transforms points from the DCS to the ICS (see equations [\(D-1\)](#) and [\(D-2\)](#)).
  - 2 – the field `transfo_matrix` specifies a matrix  $M_{ID}$  in order to transform points from the ICS to the DCS (see equation [\(D-9\)](#)).
- `transfo_matrix`: Sixteen matrix elements specified in double-precision format. The matrix elements  $m_i$  are specified from left to right and from top to bottom, as in equation [\(D-1\)](#).
- `image_color_flag`: A flag indicating color information. Color information is optional in a PIF file. If the image data is specified in an external polygonal file, this flag should be set to 0.
- 0 – there is no color information in the file.
  - 1 – the color block specifies a grey level for each array element in the data block.
  - 3 – the color block specifies RGB colors for each array element in the data block.
  - 4 – the color block specifies RGBA colors for each array element in the data block.
- `color_block_length`: Length in bytes of the color block.
- `camera_position_flag`: A flag related to the  $(x, y, z)$  position of the 3D digitizer in the DCS:
- 0 – no camera position is specified.
  - 1 – a camera position is specified by `camera_x`, `camera_y`, `camera_z`.

<code>camera_x:</code>	<code>x</code> position of the 3D digitizer in the DCS.
<code>camera_y:</code>	<code>y</code> position of the 3D digitizer in the DCS.
<code>camera_z:</code>	<code>z</code> position of the 3D digitizer in the DCS.
<code>dummy2:</code>	Reserved for future use.

### E.5.2 Block 2: 3D Data

This block contains `data_block_length` bytes of information.

- If `image_data_type` is set to 0, the data block contains a 2D array of single-precision floating-point numbers expressed in the IPICS. This data type is used to describe an interpolated parametric image. The width and height of the array are respectively equal to `array_width` and `array_height`. The array elements are written from left to right and from bottom to top. The following is the order in which the elements of a 6x4 array would be written:

```

18 19 20 21 22 23
12 13 14 15 16 17
6  7  8  9  10 11
0  1  2  3  4  5

```

The figure represents the offset of each array element in the data block.

- If `image_data_type` is set to 1, the data block contains a 2D array of  $(x, y, z)$  coordinates expressed in the DCS. This data type describes a raw parametric image. Each array element consists of three single-precision floating-point numbers, specifying respectively the  $x$ ,  $y$  and  $z$  coordinates of a measured 3D point. The array elements are written from left to right and from bottom to top.
- If `image_data_type` is set to 2, the data block contains two strings. The first string specifies the path to an external polygonal file describing the surface of the 3D image. The second string specifies the corresponding polygonal file format. A block of 512 bytes should be allocated for each string. Both strings must be written at the beginning of their block and terminated by a null character. The rest of the blocks may be filled with any characters.

### E.5.3 Block 3: Optional color information

This optional block contains `color_block_length` bytes of information. The contents of the block depend on the value of `image_color_flag`. If this flag is set to 0, the color block is empty.

- If `image_color_flag` is set to 1, the block contains a 2D array that has the same width and height as the data block array. Each array element is then 1 byte long and specifies the grey level of the corresponding vertex in the data block array.
- If `image_color_flag` is set to 3, the block contains a 2D array that has the same width and height as the data block array. Each array element is then 3 bytes long and specifies respectively the R, G, and B color components of the corresponding vertex in the data block array.
- If `image_color_flag` is set to 4, the block contains a 2D array that has the same width and height as the data block array. Each array element is then 4 bytes long and specifies respectively the R, G, B, and A color components of the corresponding vertex in the data block array.