The format of BrainVoyager UFF files

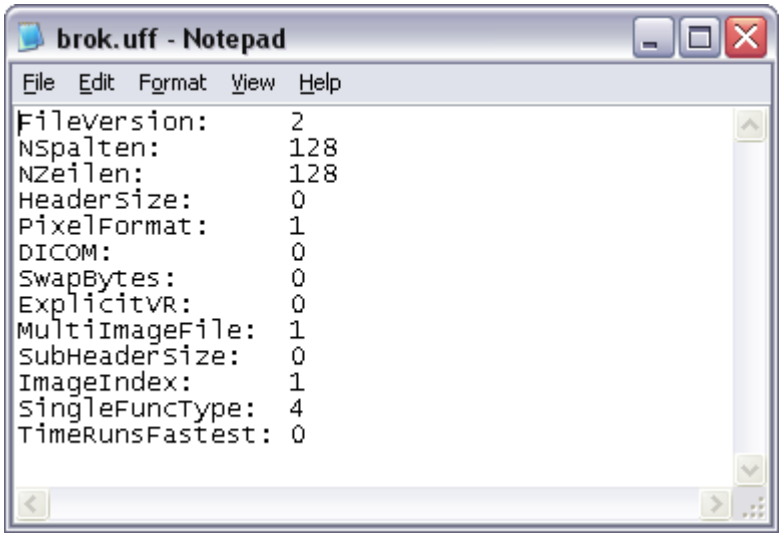
The format of BrainVoyager UFF files....1
Introduction....1
Creation of UFF files....1
Explanation of the values....3
Sample parameters....5
Matlab script to create UFF files....6

Introduction

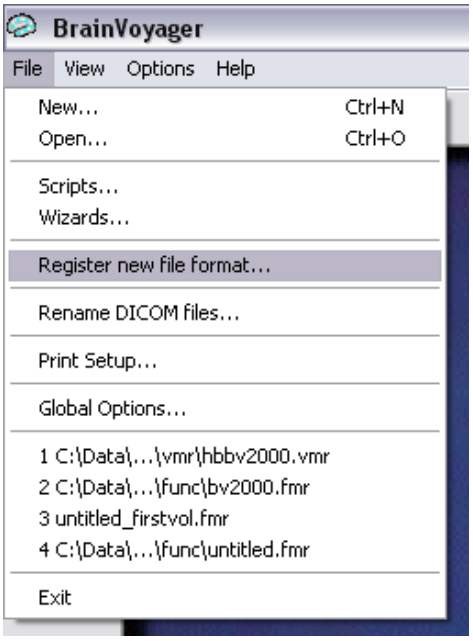
This document described the BrainVoyager user defined file formats (*.uff), which enable the BrainVoyager user to read files that are not standard recognized. The user needs to provide a specification of the header values described in the following sections. These values are saved in a *.uff text file.

Creation of UFF files

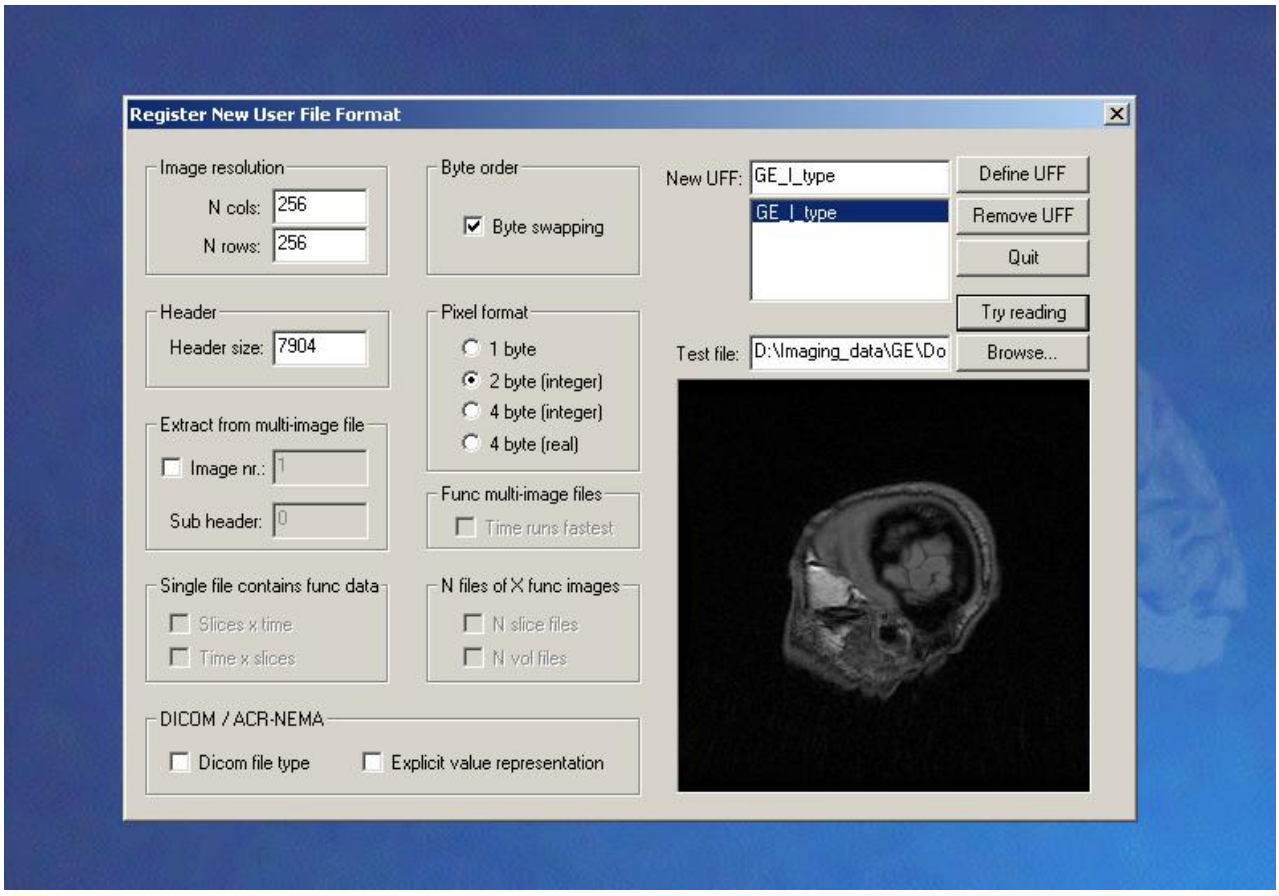
For BrainVoyager QX, the *.uff file can be created manually in a simple text editor. An example of the file is shown in the figure below. Alternatively, the Matlab script 'createUFF.m' in the last section of this document can be used to specify values via a user interface. The script will then write the *.uff file accordingly.



In BrainVoyager 2000, UFF files can be made via the "File" > "Register new file format..." function (see figure below).



In the following "Register New User File Format" dialog, not only the header values of the unknown file can be provided, but also a "Try reading" function is available, which immediately visualizes the result of the reading of the data via the provided header values on the dialog (see figure below).



Explanation of the values

Image resolution

This defines the size of a single image. 'N cols' is the image width and 'N rows' the image height. For functional (EPI) images, a common size is 64 x 64.

Header

Enter here the size of the header in number of bytes. BrainVoyager will know where to start reading for the data (intensity values).

Extract from multi-image file

In case of a multi-image file, with this function can be indicated if BrainVoyager should extract the N-th image instead of starting at the beginning from the voxel offset.

Single file contains func data

Determines whether time runs fastest (time x slices: all time points for each voxel are saved together) or whether all voxels per slice are saved together (slices x time).

N files of X func images

When there are several images in one file, indicate whether there are only images of one volume or whether there are several volumes stored in one file.

Byte order

Standard is Little Endian (Intel machines, PC compatibles), also known as LSB (Least Significant Byte) order. In case the data to read are Big Endian, check the 'Byte swapping' checkbox. Big endian is also called MSB (Most Significant Byte) order.

Pixel format

Defines in what data type the intensity values are stored. Readable data types are bytes, 16 and 32 bits integers and 32 bits real/float values.

Func multi-image files

In the case there are several images in one file and the organization of the file is so that the data are chunked together in time courses per pixel, check the 'Time runs fastest' checkbox.

When time is 'running fastest', this means that the data are saved in the following order:

Most outer loop: *Dimension Y*
Dimension X
Inner loop: *Dimension T (time)*

DICOM / ACR NEMA

When the data are saved in the DICOM format, check the 'Dicom file type' box.

Explicit value representation

Only relevant in case of DICOM files. Indicates whether the data type of the value in the value field of a DICOM 'tag' is explicitly mentioned or not.

New UFF

Write here the name of the *.uff file (without extension). This should be saved in the BrainVoyager root folder, so that BrainVoyager will find the *.uff when listing the files formats in the 'Create Project' dialog.

Sample parameters

UFF file parameter	Sample values	Description
FileVersion:	2	Describes file version of UFF format
NSpalten:	128	Number of columns
NZeilen:	64	Number of rows
HeaderSize:	0	Size of image header
PixelFormat:	1	Datatype, defining number of bytes per pixel 1 = 1 byte integer 2 = 2 bytes integer 3 = 4 bytes integer 4 = 4 bytes float
DICOM:	0	Boolean value defining whether the datatype conforms to the DICOM standard or not. 0 = no, 1 = yes.
SwapBytes:	0	Boolean value defining whether the data should be swapped or not. Default is 0 (little endian). Big endian data should have value 1.
Explicit VR:	0	Explicit value representation (only applicable in case of DICOM)
MultiImageFile:	1	Defines whether there are more than one images in the file. 0 = no, 1 = yes.
SubHeaderSize:	0	In case of a multi-image file, defines whether each image in the file does have a header (1) or not (0).
ImageIndex:	1	First image to be read from the multi-image volume
SingleFuncType:		Defines order of appearance of the dimensions in the file. 1 = slices x time (single file) 2 = time x slices (single file) 3 = N slice files 4 = N volume files
TimeRunsFastest:	0	Defines whether all timepoints per pixel or voxel appear before the next pixel or voxel values. If yes, then 'TimeRunsFastest' should be 1, 0 otherwise.

Matlab script to create UFF files

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% createUFF.m
% Purpose: create BrainVoyager user defined file format (*.uff);
% A *.uff file can be used when the image data format is not in the list of
% formats recognized by BrainVoyager automatically
%
% For activation of script: load in Matlab and press 'run'
% Hester Breman, Brain Innovation, august 2004
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

INT1BYTE = 'int8';
INT2BYTE = 'int16';
INT4BYTE = 'int32';
REAL4BYTE = 'float32';

uffImg = struct('NrOfTimepoints',{},'DimX',cell(1), 'DimY',cell(1), 'Datatype',cell(1),...
    'HeaderSize',cell(1),'DICOM',{}, 'ExplicitValueRepr',cell(1), 'MultiImgFile',{},...
    'SubHeaderSize',{},'Endianness',cell(1), 'ImageIndex',{}, ...
    'SingleFuncType',{}, 'TimeRunsFastest',{}, ...
    'Data', struct('IntensityValues',cell(1)));

ui = inputdlg('What is the number of time points in the image? (for example 180)',
'Number of time points');
uffImg(1).NrOfTimepoints = str2num(ui{1}); % get track of user input...
ui = inputdlg('What is the x dimension of the image? (for example 64)', 'X Dimension');
uffImg.DimX{1} = str2num(ui{1});
ui = inputdlg('What is the y dimension of the image? (for example 64)', 'Y Dimension');
uffImg.DimY{1} = str2num(ui{1});
dataTypes = strvcat('1 byte integer', '2 byte integer', '4 byte integer', '4 byte real');
[dType,v] = listdlg('PromptString','Select the datatype:',...
    'SelectionMode','single',...
    'ListString',dataTypes);
switch dType
    case 1
        uffImg.Datatype{1} = INT1BYTE; % in *.uff file: 0
    case 2
        uffImg.Datatype{1} = INT2BYTE; % in *.uff file: 1
    case 3
        uffImg.Datatype{1} = INT4BYTE; % in *.uff file: 2
    case 4
        uffImg.Datatype{1} = REAL4BYTE; % in *.uff file: 3
end
button = questdlg('Is the data in the DICOM format?','DICOM compliance','Yes','No','No');
if strcmp(button,'Yes')
    uffImg(1).DICOM = 1;
    button = questdlg('Do the data have an explicit value
representation?','Explicit','Yes','No','No');
    if strcmp(button,'Yes')
        uffImg.ExplicitValueRepr{1} = 1;
    else
        uffImg.ExplicitValueRepr{1} = 0;
    end
else
    uffImg(1).DICOM = 0;
    uffImg.ExplicitValueRepr{1} = 0;
end
button = questdlg('Are the data little or big endian?','Endianness','Little Endian','Big
Endian','Little Endian');
if strcmp(button,'Little Endian')
    uffImg.Endianness{1} = 0;
else
    uffImg.Endianness{1} = 1; % swap bytes
end
ui = inputdlg('What is the header size of the image? (for example 0)', 'Header size');
uffImg.HeaderSize{1} = str2num(ui{1});
button = questdlg('Are there multiple images in one file?','Multi Image
File','Yes','No','No');
```

```
if strcmp(button,'Yes')
    uffImg(1).MultiImgFile = 1;
    ui = inputdlg('What is the size of the subheader of each image? (for example 0)',
'Subheader size');
    uffImg(1).SubHeaderSize = str2num(ui{1});
    ui = inputdlg('What is the first image in the file to be read? (for example 1)',
'Image index');
    uffImg(1).ImageIndex = str2num(ui{1});
    singleFuncType = strvcat('slices x time', 'time x slices', 'N slice files of X
functional images', 'N volume files of X functional images');
    [funcType,v] = listdlg('PromptString','Please choose the order of appearance of the
dimensions in the file:',...
        'SelectionMode','single',...
        'ListString',singleFuncType);
    uffImg(1).SingleFuncType = funcType; %1 = slices x time 2 = time x slices 3 = N slice
files 4 = N vol files
    button = questdlg('Do all timepoints per pixel/voxel appear before the next
pixel/voxel values?','Time runs fastest','Yes','No','No');
    if strcmp(button,'Yes')
        uffImg(1).TimeRunsFastest = 1;
    else
        uffImg(1).TimeRunsFastest = 0;
    end
else
    uffImg(1).MultiImgFile = 0;
    uffImg(1).SubHeaderSize = 0;
    uffImg(1).ImageIndex = 1;
    uffImg(1).SingleFuncType = 0;
    uffImg(1).TimeRunsFastest = 0;
end

[FileName,PathName] = uigetfile('*.*','Please select the image file');
imgFile = strcat(PathName, FileName);
if uffImg(1).DICOM == 1
    uffImg.Data.IntensityValues{i} = dicomread(imgFile);
else
    fid = fopen(imgFile,'rb');
    if uffImg(1).MultiImgFile == 0
        for i=1:uffImg(1).NrOfTimepoints
            try
                uffImg.Data.IntensityValues{i} =
fread(fid,[uffImg.DimX{1}*uffImg.DimY{1}],uffImg.Datatype{1});
            catch
                disp(lasterr);
            end
        end
    end
    fclose(fid);
end
[pathstr,name,ext,versn] = fileparts(imgFile);
uffName = [PathName name '.uff'];
fid = fopen(uffName,'w');
fprintf(fid, '%s\t%s\n', 'FileVersion:','2');
fprintf(fid, '%s\t\t%s\n', 'NSpalten:',num2str(uffImg.DimX{1})); % nr of columns
fprintf(fid, '%s\t\t%s\n', 'NZeilen:', num2str(uffImg.DimY{1})); % nr of rows
fprintf(fid, '%s\t\t%s\n','HeaderSize:', num2str(uffImg.HeaderSize{1}));
fprintf(fid, '%s\t%s\n','PixelFormat:', num2str(dType-1));
fprintf(fid, '%s\t\t%s\n','DICOM:',num2str(uffImg(1).DICOM));
fprintf(fid, '%s\t\t%s\n','SwapBytes:', num2str(uffImg.Endianness{1}));
fprintf(fid, '%s\t\t%s\n','ExplicitVR:', num2str(uffImg.ExplicitValueRepr{1}));
fprintf(fid, '%s\t%s\n','MultiImageFile:', num2str(uffImg(1).MultiImgFile));
fprintf(fid, '%s\t%s\n','SubHeaderSize:', num2str(uffImg(1).SubHeaderSize));
fprintf(fid, '%s\t\t%s\n','ImageIndex:', num2str(uffImg(1).ImageIndex));
fprintf(fid, '%s\t%s\n','SingleFuncType:', num2str(uffImg(1).SingleFuncType));
fprintf(fid, '%s\t%s\n','TimeRunsFastest:', num2str(uffImg(1).TimeRunsFastest));
fclose(fid);

% show
if uffImg(1).MultiImgFile == 0 && uffImg(1).DICOM == 0
    img = uffImg.Data.IntensityValues{1};
    img2D = zeros(uffImg.DimX{1},uffImg.DimY{1});
```

```
    img2D(:) = img(:);
    colormap(gray)
    imagesc(img2D)
end

msgbox(strcat('Ready! *.uff file can be found in directory: ',PathName), 'End of script');
```