Aug 07, 2025

Processing Raw 2D Cine MR Data from Elekta Unity Using MATLAB

Scientific Reports

DOI
https://dx.doi.org/10.17504/protocols.io.ewov11oq7vr2/v1

Jiwon Sung[1], Jihun Kim[1]

[1]Gangnam Severance Hospital

지원 Jiwon Sung
Gangnam Severance Hospital

Create & collaborate more with a free account

Edit and publish protocols, collaborate in communities, share insights through comments, and track progress with run records.

Create free account

OPEN ACCESS

DOI: https://dx.doi.org/10.17504/protocols.io.ewov11oq7vr2/v1

External link: https://doi.org/10.1038/s41598-025-15107-4

Protocol Citation: Jiwon Sung, Jihun Kim 2025. Processing Raw 2D Cine MR Data from Elekta Unity Using MATLAB. **protocols.io** https://dx.doi.org/10.17504/protocols.io.ewov11oq7vr2/v1

Manuscript citation:
Sung J, Choi Y, Kim J, Kim JW, Kim J (2025) Development of in-house software to process real-time cine magnetic resonance images acquired during 1.5 T MR-guided radiation therapy. Scientific Reports 15(). doi: 10.1038/s41598-025-15107-4

License: This is an open access protocol distributed under the terms of the **Creative Commons Attribution License**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Protocol status: Working
We use this protocol and it's working

Created: August 06, 2025

Last Modified: August 07, 2025

Protocol Integer ID: 224221

Keywords: processing raw 2d cine mr data, raw 2d cine mr data from elekta unity, raw binary files of 2d cine mr image, 2d cine mr image, motion analysis during mr, readable medical image format, converted image, guided radiotherapy, linac system, using matlab, elekta unity mr, raw binary file, classification by anatomical plane, house matlab, conversion into mha, matlab this protocol, dicom format

Abstract

This protocol describes an in-house MATLAB-based pipeline to convert raw binary files of 2D cine MR images, acquired via the Treatment Session Manager (TSM) on the Elekta Unity MR-Linac system, into readable medical image formats. The software includes automatic renaming, classification by anatomical planes (axial, sagittal, coronal), and conversion into MHA and DICOM formats. The converted images can be used for quantitative geometric and motion analysis during MR-guided radiotherapy (MRgRT).

Guidelines

GetMotionMonitoringImage.m GetMotionMonitoringInfo.m writemha2D.m

Main MatLAB script

```
clear all
close all
clc

%% change the  binary files name

WorkingFolder = input('Write the path of patient folder:\n\nex)
E:\MM\1.3.\ ....\n\n:','s')
folder_TwoDImages = sprintf('%s\\TwoDImages',WorkingFolder);
binaryfiles = dir(sprintf('%s\\Frame_ID_*.bin',folder_TwoDImages));

for ff = 1:size(binaryfiles)
   filename = binaryfiles(ff).name;

   strsplit_filename = strsplit(filename,'_');
   time_info = strsplit_filename{4};

   filepath = sprintf('%s\\%s',binaryfiles(ff).folder,filename);
   filepath_rep = strrep(filepath,'\','\\');

   command_wMIC = sprintf('wmic datafile where name="%s" get lastmodified |
findstr /brc:[0-9]',filepath_rep);
   [~, modifiedTime] = system(command_wMIC);

   str_date = modifiedTime(1:8);
   str_time = modifiedTime(9:14);
   str_microseconds = modifiedTime(16:21);

   filename_renamed =
sprintf('file_%s_%s_%s_%s.bin',str_date,str_time,str_microseconds,time_info);
   filepath_renamed = sprintf('%s',filename_renamed);

   command = sprintf('rename "%s" "%s"',filepath,filepath_renamed);
   system(command);
end
```

```
%% Detection of Discontinuous Time Intervals

binaryfiles = dir(sprintf('%s\\file_*.bin',folder_TwoDImages));
acquisition_timepoints = zeros(size(binaryfiles,1),1);

for ff = 1:size(binaryfiles)
   filename = binaryfiles(ff).name;

   strsplit_filename = strsplit(filename,'_');

   str_hhmmss = strsplit_filename{3};
   str_ms = strsplit_filename{4};

   hh = str2double(str_hhmmss(1:2));
   mm = str2double(str_hhmmss(3:4));
   ss = str2double(str_hhmmss(5:6));
   ms = str2double(str_ms);      % in microsecond

   % calculate timepoint in milliseconds
   timestamp_in_millisecond = 60*60*1000*hh + 60*1000*mm + 1000*ss + ms/1000;

   acquisition_timepoints(ff,1) = timestamp_in_millisecond;
end

acquisition_timeinterval = abs(diff(acquisition_timepoints));

[max_interval, index] = max(acquisition_timeinterval);
[maximum5intervals, index1] = maxk(acquisition_timeinterval,5);

aa= maximum5intervals > 10^5;
aaIndex = find(aa==1);
aaRealIndex=index1(aaIndex);

if max_interval > 10^5
   filename_discontinuity = sprintf('%s\\discontinuity.txt',folder_TwoDImages);
   fid = fopen(filename_discontinuity,'w');
   for bb = 1:length(aaRealIndex)
      filename_temp = binaryfiles(aaRealIndex(bb)).name;
      strsplit_filename = strsplit(filename_temp,'_');
      str_yyyymmdd = strsplit_filename{2};
      str_hhmmss = strsplit_filename{3};
      str_ms = strsplit_filename{4};
      fprintf(fid,'%s_%s_%s\n',str_yyyymmdd,str_hhmmss,str_ms);
```

```
    end
    fclose(fid);
end



if exist ('filename_discontinuity','var')
    warning('it is interval in here, Please separate the MM files btw intervals')
    pause
end

%% Classification of axial coronal sagittal

binaryfiles = dir(sprintf('%s\\file_*.bin',folder_TwoDImages));
[~,~,~,ImageSizeRow,ImageSizeColumn]= GetMotionMonitoringInfo(WorkingFolder);
phantom_ID = zeros(ImageSizeRow, size(binaryfiles,1));

for ff=1:size(binaryfiles)
    filename_temp = binaryfiles(ff).name;
    fprintf('Fraction: %s\n',filename_temp)
    foldername_temp = binaryfiles(ff).folder;
    filepath_temp = sprintf('%s\\%s',foldername_temp, filename_temp);
    mm_image = GetMotionMonitoringImage(filepath_temp, WorkingFolder);



    image(mm_image)
    phantom_ID(:,ff)=mm_image(1:ImageSizeRow,1);
end

ImageSizeRow_axial=phantom_ID(ImageSizeRow-5,:);
ImageSizeRow_sagittal=phantom_ID(ImageSizeRow-13,:);
ImageSizeRow_coronal=phantom_ID(ImageSizeRow-9,:);
ImageSizeRow_axial_factors=unique(ImageSizeRow_axial);
ImageSizeRow_sagittal_factors=unique(ImageSizeRow_sagittal);
ImageSizeRow_coronal_factors=unique(ImageSizeRow_coronal);
if length(ImageSizeRow_sagittal_factors) > 2
warning('separate the binary files')
return;
elseif length(ImageSizeRow_coronal_factors) > 2
warning('separate the binary files')
return;
elseif length(ImageSizeRow_axial_factors) > 2
warning('separate the binary files')
return;
```

```
end
if sum(ImageSizeRow_axial_factors(1,1)==ImageSizeRow_axial) >
sum(ImageSizeRow_axial_factors(1,2)==ImageSizeRow_axial)
axialFactor=ImageSizeRow_axial_factors(1,2)
else
axialFactor=ImageSizeRow_axial_factors(1,1)
end
if sum(ImageSizeRow_coronal_factors(1,1)==ImageSizeRow_coronal) >
sum(ImageSizeRow_coronal_factors(1,2)==ImageSizeRow_coronal)
coronalFactor=ImageSizeRow_coronal_factors(1,2)
else
coronalFactor=ImageSizeRow_coronal_factors(1,1)
end
if sum(ImageSizeRow_sagittal_factors(1,1)==ImageSizeRow_sagittal) >
sum(ImageSizeRow_sagittal_factors(1,2)==ImageSizeRow_sagittal)
sagittalFactor=ImageSizeRow_sagittal_factors(1,2)
else
sagittalFactor=ImageSizeRow_sagittal_factors(1,1)
end
folder_TwoDImagesRenamed_axial = sprintf('%s\\axial',folder_TwoDImages);
folder_TwoDImagesRenamed_sagittal = sprintf('%s\\sagittal',folder_TwoDImages);
folder_TwoDImagesRenamed_coronal = sprintf('%s\\coronal',folder_TwoDImages);
if ~exist(folder_TwoDImagesRenamed_axial,'dir')
mkdir(folder_TwoDImagesRenamed_axial)
end
if ~exist(folder_TwoDImagesRenamed_sagittal,'dir')
mkdir(folder_TwoDImagesRenamed_sagittal)
end
if ~exist(folder_TwoDImagesRenamed_coronal,'dir')
mkdir(folder_TwoDImagesRenamed_coronal)
end
for ff=1:size(binaryfiles)
filename_temp = binaryfiles(ff).name;
fprintf('Fraction: %s\n',filename_temp)
foldername_temp = binaryfiles(ff).folder;
filepath_temp = sprintf('%s\\%s',foldername_temp, filename_temp);
   mm_image = GetMotionMonitoringImage(filepath_temp, WorkingFolder);
   if mm_image(ImageSizeRow-9,1) == coronalFactor
      folder_to_sort = sprintf('%s\\coronal',folder_TwoDImages);
      filepath_new = sprintf('%s\\%s',folder_to_sort,filename_temp);
      movefile(filepath_temp,filepath_new)
   elseif mm_image(ImageSizeRow-13,1) == sagittalFactor
      folder_to_sort = sprintf('%s\\sagittal',folder_TwoDImages);
```

```
        filepath_new = sprintf('%s\\%s',folder_to_sort,filename_temp);
        movefile(filepath_temp,filepath_new)
    elseif mm_image(ImageSizeRow-5,1) == axialFactor
        folder_to_sort = sprintf('%s\\axial',folder_TwoDImages);
        filepath_new = sprintf('%s\\%s',folder_to_sort,filename_temp);
        movefile(filepath_temp,filepath_new)
    end

end

%%  Convert binary files to MHA files

ListPlanes = {'coronal','sagittal','axial'};
offset = [0.0, 0.0, 0.0];

% image information
[SliceDimensionXInmm, SliceDimensionYInmm, SliceDimensionZInmm, Rows, Columns] =
GetMotionMonitoringInfo(WorkingFolder);

spacing = zeros(1,3);
spacing(1) = SliceDimensionXInmm/Rows;
spacing(2) = SliceDimensionYInmm/Columns;
spacing(3) = SliceDimensionZInmm;

% folder containing two-dimensional motion monitoring images
folder_TwoDImages_mha = sprintf('%s\\TwoDImages_mha',folder_TwoDImages);
folder_TwoDImages_mha_axi =
sprintf('%s\\TwoDImages_mha\\axial',folder_TwoDImages);
folder_TwoDImages_mha_sag =
sprintf('%s\\TwoDImages_mha\\sagittal',folder_TwoDImages);
folder_TwoDImages_mha_cor =
sprintf('%s\\TwoDImages_mha\\coronal',folder_TwoDImages);

if ~exist(folder_TwoDImages_mha,'dir')
    mkdir(folder_TwoDImages_mha)
end
if ~exist(folder_TwoDImages_mha_axi,'dir')
    mkdir(folder_TwoDImages_mha_axi)
end
if ~exist(folder_TwoDImages_mha_sag,'dir')
    mkdir(folder_TwoDImages_mha_sag)
end
if ~exist(folder_TwoDImages_mha_cor,'dir')
```

```
    mkdir(folder_TwoDImages_mha_cor)
end

for plane = 1:3
    folder_TwoDImages_plane =
sprintf('%s\\%s',folder_TwoDImages,ListPlanes{plane});
    binaryfiles = dir(sprintf('%s\\file_*.bin',folder_TwoDImages_plane));

    for ff = 1:size(binaryfiles,1)
        filename_bn = binaryfiles(ff).name;
        filepath_bn = sprintf('%s\\%s',binaryfiles(ff).folder,filename_bn);

        % folder for mha file
        folder_TwoDImages_mha_plane =
sprintf('%s\\%s',folder_TwoDImages_mha,ListPlanes{plane});
        filepath_mha =
sprintf('%s\\img_%s.mha',folder_TwoDImages_mha_plane,filename_bn(6:end-4));

        mm_image_2D = GetMotionMonitoringImage(filepath_bn, WorkingFolder);

        mm_image_3D = zeros([size(mm_image_2D), 1]);
        mm_image_3D(:,:,1) = mm_image_2D;

         writemha2D(filepath_mha,mm_image_3D,offset,spacing,'ushort');
    end
end

%% Convert MHA files to DCM

% MM center
[MMcenterX, MMcenterY, MMcenterZ]  = GetMMcenterInfo(WorkingFolder);

ListPlanes = {'coronal','sagittal','axial'};
PatientID = input('Patient ID: ','s')
PatientName = input('Patient name: ','s')

for plane = 1:3
    input_folder = sprintf('%s\\%s',folder_TwoDImages_mha, ListPlanes{plane});
    output_folder = sprintf('%s\\TwoDImages_dcm\\%s',folder_TwoDImages,
ListPlanes{plane});

    if ~exist(output_folder,'dir')
        mkdir(output_folder)
```

```
end

files = dir(sprintf('%s\\*.mha',input_folder));

for ff = 1:size(files,1)
    filename_mha = sprintf('%s\\%s',files(ff).folder,files(ff).name);

    % read mha files
    image_header = mha_read_header(filename_mha);
    image_origin = image_header.Offset;
    image_spacing = image_header.PixelDimensions;
    image_size = image_header.Dimensions;
    image = mha_read_volume(image_header);
    image = permute(image,[2 1 3]);

    % write dicom files
    filename_temp = files(ff).name;
    filename_dcm = sprintf('%s\\%s.dcm',output_folder,filename_temp(1:end-4));

    dicomwrite(image,filename_dcm)

    % write dicom information
    info = dicominfo(filename_dcm);
    if ff == 1
        SeriesInstanceUID = info.SeriesInstanceUID;
    else
        info.SeriesInstanceUID = SeriesInstanceUID;
    end

    info.PatientID =  PatientID;
    info.PatientName = PatientName;
    info.PixelSpacing = image_spacing(1:2);
    info.SliceThickness = image_spacing(3);
    info.Modality = 'MR';
    if plane == 1
        info.SeriesDescription = 'CORONAL';
        info.ImageType = 'PRIMARY\CORONAL';
        info.ImageOrientationPatient = [1, 0, 0, 0, 0, -1];
        ImagePositionPatient = zeros(1,3);
        ImagePositionPatient(1) =  -image_spacing(1)*(image_size(1) -
1)/2+MMcenterX;
        ImagePositionPatient(2) =  -MMcenterZ;
        ImagePositionPatient(3) =  image_spacing(1)*(image_size(1) - 1)/2
```

```
MMcenterY;
        info.ImagePositionPatient = ImagePositionPatient;
    elseif plane == 2
        info.SeriesDescription = 'SAGITTAL';
        info.ImageType = 'PRIMARY\SAGITTAL';
        info.ImageOrientationPatient = [0, 1, 0, 0, 0, -1];
        ImagePositionPatient = zeros(1,3);
        ImagePositionPatient(1) =  MMcenterY;
        ImagePositionPatient(2) =   -image_spacing(1)*(image_size(1) - 1)/2
MMcenterZ;
        ImagePositionPatient(3) =   image_spacing(1)*(image_size(1) -
1)/2+MMcenterX;
        info.ImagePositionPatient = ImagePositionPatient;
    elseif plane == 3
        info.SeriesDescription = 'AXIAL';
        info.ImageType = 'PRIMARY\AXIAL';
        info.ImageOrientationPatient = [1, 0, 0, 0, 1, 0];
        ImagePositionPatient = zeros(1,3);
        ImagePositionPatient(1) = -image_spacing(1)*(image_size(1) -
1)/2+MMcenterX;
        ImagePositionPatient(2) = -image_spacing(1)*(image_size(1) -
1)/2+MMcenterY;
        ImagePositionPatient(3) =  -MMcenterZ;
        info.ImagePositionPatient = ImagePositionPatient;

    end

    dicomwrite(image,filename_dcm,info,"CreateMode","copy");

    info_read = dicominfo(filename_dcm);

  end
end

%% Coordinate for each plane

MMcenter = [MMcenterX, MMcenterY, MMcenterZ]
MMaxial = [MMcenterX, MMcenterY, -MMcenterZ]
MMsagittal = [MMcenterY, -MMcenterZ, MMcenterX]
MMcoronal = [MMcenterX, -MMcenterZ, -MMcenterY]
```

Materials

1. Matlab (MathWorks, Natick, USA).
2. 2D cine MR data acquired from Elekta Unity (Elekta AB, Stockholm, Sweden)

- Binary files in TwoDImages folder
- MotionMonitoring2DImages.ExamCardInfo in ExamCards folder
- MM information data in BinaryMasks folder

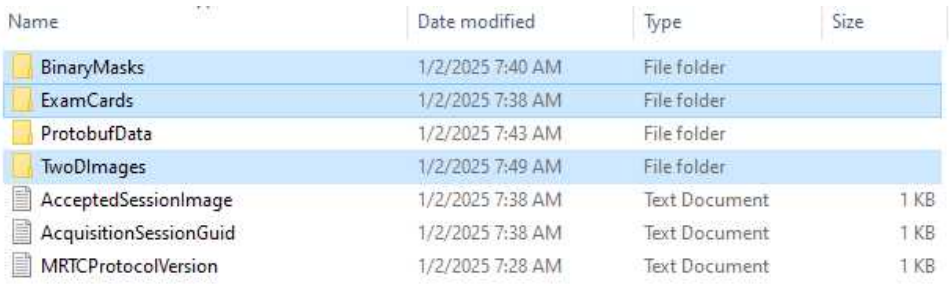

Figure 1. 2D cine MR data needed for this protocol

Step 1: Input Patient Folder Path and Rename Binary Files

1 **1-1. Prompt the user to input the patient folder path**
When the script is executed, it prompts the user to manually enter the directory path where the patient data is stored.

1-2. Identify all raw binary files with the format 'Frame_ID_*.bin'
The script locates all binary files in the TwoDImages subdirectory of the specified folder that follow the naming pattern Frame_ID_*.bin.

1-3. Extract timestamp metadata from each file
Using the Windows wmic command, the script retrieves the last modified date, time, and microseconds for each file.

1-4. Generate a standardized filename for each binary file
Each file is renamed using the following standardized format:
file_YYYYMMDD_HHMMSS_MICROSEC_TIMEINFO.bin

Step 2: Detection of Discontinuous Time Intervals

2 **2-1. Extract time information from each binary file name**
The script parses each renamed file to extract its acquisition time (HHMMSS and microseconds) from the filename.

2-2. Convert to absolute timestamps (in milliseconds)
Each timepoint is converted into milliseconds for uniform comparison between files.

2-3. Calculate time intervals between consecutive files
The difference between consecutive timestamps is computed. If the interval exceeds a defined threshold (e.g., 100,000 ms), it may indicate an acquisition gap.

2-4. Save list of discontinuities (if any)
If large time gaps are detected, the filenames of those frames are saved in a text file named discontinuity.txt for user review.
⇒ It is necessary to delete files that have time intervals.

Step 3: Classify Images by Anatomical Plane

3 The classification algorithm requires the complete set of planes (coronal, sagittal, and axial) to function properly.

3-1. Load each binary file as a 2D image

The script uses a custom function to convert the raw binary files into 2D matrices.

3-2. Extract reference rows for plane classification
Specific pixel rows are analyzed to identify unique patterns associated with axial, sagittal, or coronal slices.

3-3. Automatically assign image orientation
Each file is assigned to one of the three planes (axial, sagittal, coronal) based on pattern-matching.

3-4. Move files into corresponding folders
Classified images are moved to:

- TwoDImages\axial
- TwoDImages\sagittal
- TwoDImages\coronal

Step 4: Convert Binary Files to MHA Format

4 **4-1. Load image metadata from JSON ExamCard**
The script retrieves image dimensions and pixel spacing from the MotionMonitoring2DImages.ExamCardInfo.json file.

4-2. Convert each 2D slice into a 3D volume
Each 2D image is converted into a 3D matrix to fit the MHA (MetaImage) format.

4-3. Save MHA files with appropriate spacing and offset
The MHA files are saved with correct voxel spacing and offsets in the following directories:

- TwoDImages_mha\axial
- TwoDImages_mha\sagittal
- TwoDImages_mha\coronal

Step 5: Convert MHA Files to DICOM Format

5 **5-1. Prompt user for Patient ID and Name**
The script asks the user to input patient information, which is embedded in the DICOM header.

5-2. Read and permute image data
Each MHA file is read and permuted to match the standard DICOM orientation (rows/columns).

5-3. Assign orientation and position per plane

- Coronal: [1 0 0, 0 0 -1]
- Sagittal: [0 1 0, 0 0 -1]
- Axial: [1 0 0, 0 1 0]

5-4. Write DICOM files with proper tags

DICOM files are saved with:

- Patient info
- Pixel spacing
- Image orientation & position
- Series description

Saved in:

- TwoDImages_dcm\axial
- TwoDImages_dcm\sagittal
- TwoDImages_dcm\coronal