THUNDER

Generated by Doxygen 1.8.13

Contents

1	Clas	s Index	Ĭ.													1
	1.1	Class	List					 	 	 	 		 			1
2	File	Index														3
	2.1	File Lis	st					 	 	 	 		 		 	3
3	Clas	s Docu	mentation	1												5
	3.1	MRCH	leader Stru	ıct Refe	rence			 	 	 	 		 		 	5
		3.1.1	Detailed	Descrip	otion			 	 	 	 		 		 	6
		3.1.2	Member	Data Do	ocume	ntation	n	 	 	 	 		 		 	6
			3.1.2.1	cella				 	 	 	 		 		 	6
			3.1.2.2	cellb				 	 	 	 		 		 	6
			3.1.2.3	dmax				 	 	 	 		 			6
			3.1.2.4	dmea	n			 	 	 	 		 		 	6
			3.1.2.5	dmin				 	 	 	 		 		 	6
			3.1.2.6	extra				 	 	 	 		 		 	6
			3.1.2.7	ispg.				 	 	 	 		 		 	7
			3.1.2.8	label				 	 	 	 		 		 	7
			3.1.2.9	machs	st			 	 	 	 		 		 	7
			3.1.2.10	map				 	 	 	 		 			7
			3.1.2.11	mapc				 	 	 	 		 			7
			3.1.2.12	mapr				 	 	 	 		 		 	7
			3.1.2.13	maps				 	 	 	 		 		 	7
			31214	mode												8

ii CONTENTS

			3.1.2.15	mx	8
			3.1.2.16	my	8
			3.1.2.17	mz	8
			3.1.2.18	nlabels	8
			3.1.2.19	nsymbt	8
			3.1.2.20	nx	8
			3.1.2.21	nxstart	9
			3.1.2.22	ny	9
			3.1.2.23	nystart	9
			3.1.2.24	nz	9
			3.1.2.25	nzstart	9
			3.1.2.26	origin	9
			3.1.2.27	rms	9
4	File	Docume	entation		11
	4.1			//Euler.h File Reference	11
		4.1.1		Description	12
		4.1.2		Documentation	13
			4.1.2.1	alignZ()	13
			4.1.2.2	angle() [1/3]	13
			4.1.2.3	angle() [2/3]	14
			4.1.2.4	angle() [3/3]	14
			4.1.2.5	direction()	14
			4.1.2.6	quaternion() [1/3]	16
			4.1.2.7	quaternion() [2/3]	16
			4.1.2.8	quaternion() [3/3]	17
			4.1.2.9	quaternion conj()	17
					17
				quaternion_mul()	
				randRotate2D()	18
				randRotate3D()	18
			4.1.2.13	reflect3D()	18

CONTENTS

		4.1.2.14	rotate2D() [1/2]	. 19
		4.1.2.15	rotate2D() [2/2]	. 19
		4.1.2.16	rotate3D() [1/3]	. 19
		4.1.2.17	rotate3D() [2/3]	. 20
		4.1.2.18	rotate3D() [3/3]	. 20
		4.1.2.19	rotate3DX()	. 20
		4.1.2.20	rotate3DY()	. 21
		4.1.2.21	rotate3DZ()	. 21
		4.1.2.22	swingTwist()	. 21
4.2	include	e/Geometry	y/Transformation.h File Reference	. 22
	4.2.1	Detailed	Description	. 22
	4.2.2	Function	Documentation	. 23
		4.2.2.1	SYMMETRIZE_FT()	. 23
		4.2.2.2	SYMMETRIZE_RL()	. 23
		4.2.2.3	VOL_TRANSFORM_MAT_FT()	. 24
		4.2.2.4	VOL_TRANSFORM_MAT_RL()	. 24
4.3	include	e/Image/MF	RCHeader.h File Reference	. 25
	4.3.1	Detailed	Description	. 25
Index				27

Chapter 1

Class Index

a.	4	_	1	
Т.	П.	(:	lace	ICT

Here are the classes,	structs, unions	and interfaces	with brief descr	iptions:
,	,			•

MRCHeader	
-----------	--

2 Class Index

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

include/Geometry/Euler.h	
Euler.h contains several functions, for operations of quaternions, converting between Euler an-	
gles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution	11
include/Geometry/Transformation.h	
Transformation.h contains several functions, for transformation of volume according to symmetry	22
include/Image/MRCHeader.h	
MRCHeader.h contains the main header of the MRC format, including fixed format values for metadata about the images/volumes	25

File Index

Chapter 3

Class Documentation

3.1 MRCHeader Struct Reference

The constitution of MRC main header.

```
#include <MRCHeader.h>
```

Public Attributes

- int nx
- int ny
- int nz
- int mode
- int nxstart
- int nystart
- · int nzstart
- int mx
- int my
- int mzfloat cella [3]
- float cellb [3]
- int mapc
- int mapr
- int maps
- float dmin
- float dmax
- float dmean
- int ispg
- int nsymbt
- char extra [100]
- float origin [3]
- char map [4]
- int machst
- float rms
- int nlabels
- char label [10][80]

6 Class Documentation

3.1.1 Detailed Description

The constitution of MRC main header.

The main header is limited to 1024 bytes, but includes unassigned space in anticipation of future extensions.

3.1.2 Member Data Documentation

extra space used for anything - 0 by default

```
3.1.2.1 cella
float MRCHeader::cella[3]
cell dimensions in angstroms
3.1.2.2 cellb
float MRCHeader::cellb[3]
cell angles in degrees
3.1.2.3 dmax
float MRCHeader::dmax
maximum density value
3.1.2.4 dmean
float MRCHeader::dmean
mean density value
3.1.2.5 dmin
float MRCHeader::dmin
minimum density value
3.1.2.6 extra
char MRCHeader::extra[100]
```

```
3.1.2.7 ispg
int MRCHeader::ispg
space group number 0 or 1 (default=0)
3.1.2.8 label
char MRCHeader::label[10][80]
ten 80-character text labels Symmetry records follow - if any - stored as text as in International Tables, operators
separated by * and grouped into 'lines' of 80 characters (ie. symmetry operators do not cross the ends of the
80-character 'lines' and the 'lines' do not terminate in a *). Data records follow.
3.1.2.9 machst
int MRCHeader::machst
machine stamp
3.1.2.10 map
char MRCHeader::map[4]
character string 'MAP' to identify file type
3.1.2.11 mapc
int MRCHeader::mapc
axis corresp to cols (1,2,3 for X,Y,Z)
3.1.2.12 mapr
int MRCHeader::mapr
axis corresp to rows (1,2,3 for X,Y,Z)
3.1.2.13 maps
int MRCHeader::maps
```

axis corresp to sections (1,2,3 for X,Y,Z)

8 Class Documentation

3.1.2.14 mode int MRCHeader::mode $data\ type->0\ -\ image:\ signed\ 8-bit\ bytes\ range\ -128\ to\ 127;\ 1\ -\ image:\ 16-bit\ halfwords;\ 2\ -\ image:\ 32-bit\ reals;$ 3 - transform : complex 16-bit integers; 6 - image : unsigned 16-bit range 0 to 65535 3.1.2.15 mx int MRCHeader::mx number of intervals along X 3.1.2.16 my int MRCHeader::my number of intervals along Y 3.1.2.17 mz int MRCHeader::mz number of intervals along Z 3.1.2.18 nlabels int MRCHeader::nlabels number of labels being used 3.1.2.19 nsymbt int MRCHeader::nsymbt number of bytes used for symmetry data (0 or 80) 3.1.2.20 nx int MRCHeader::nx

number of columns (fastest changing in map)

```
3.1.2.21 nxstart
int MRCHeader::nxstart
number of first column in map (Default = 0)
3.1.2.22 ny
int MRCHeader::ny
number of rows
3.1.2.23 nystart
int MRCHeader::nystart
number of first row in map
3.1.2.24 nz
int MRCHeader::nz
number of sections (slowest changing in map)
3.1.2.25 nzstart
int MRCHeader::nzstart
number of first section in map
3.1.2.26 origin
float MRCHeader::origin[3]
origin in X,Y,Z used for transforms
3.1.2.27 rms
float MRCHeader::rms
rms deviation of map from mean density
The documentation for this struct was generated from the following file:
```

Generated by Doxygen

• include/Image/MRCHeader.h

10 Class Documentation

Chapter 4

File Documentation

4.1 include/Geometry/Euler.h File Reference

Euler.h contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution.

```
#include <cmath>
#include <gsl/gsl_math.h>
#include "Macro.h"
#include "Typedef.h"
#include "Precision.h"
#include "Random.h"
#include "Functions.h"
```

Functions

• void quaternion_mul (dvec4 &dst, const dvec4 &a, const dvec4 &b)

Calculate the product of two quaternions q_1 and q_2 .

dvec4 quaternion_conj (const dvec4 &quat)

Calculate the conjugate quaternion of a quaternion.

• void angle (double &phi, double &theta, const dvec3 &src)

Calculate ϕ and θ given a certain direction \mathbf{v} .

• void angle (double &phi, double &theta, double &psi, const dmat33 &src)

Calculate ϕ,θ and ψ of the rotation represented by the rotation matrix R.

void angle (double &phi, double &theta, double &psi, const dvec4 &src)

Calculate ϕ , θ and ψ of the rotation represented by the unit quaternion \mathbf{q} .

• void quaternion (dvec4 &dst, const double phi, const double theta, const double psi)

Calculate the unit quaternion \mathbf{q} for representing the rotation, given 3 Euler angles ϕ , θ and ψ .

void quaternion (dvec4 &dst, const double phi, const dvec3 &axis)

Calculate the unit quaternion \mathbf{q} for representing the rotation, given the rotation axis \mathbf{r} and the rotation angle around this axis ϕ .

• void quaternion (dvec4 &dst, const dmat33 &src)

Calculate the unit quaternion ${\bf q}$ for representing the rotation, given the rotation matrix ${\bf R}$.

void rotate2D (dmat22 &dst, const dvec2 &vec)

Calculate the rotation matrix (2D) \mathbf{R} , which rotates the unit vector $\mathbf{v_0} = \{1, 0\}$ to the given unit vector \mathbf{v} .

void rotate2D (dmat22 &dst, const double phi)

Calculate the rotation matrix (2D) \mathbf{R} , given the rotation angle ϕ .

• void direction (dvec3 &dst, const double phi, const double theta)

Caclulate the unit direction vector \mathbf{v} , given the rotation angle ϕ and θ .

void rotate3D (dmat33 &dst, const double phi, const double theta, const double psi)

Caclulate the rotation matrix \mathbf{R} , given the rotation angle ϕ , θ and ψ .

void rotate3D (dmat33 &dst, const dvec4 &src)

Calculate the rotation matrix \mathbf{R} , given the unit quaternion \mathbf{q} which represents this rotation.

void rotate3DX (dmat33 &dst, const double phi)

Calculate the rotation matrix \mathbf{R} which represents the rotation along X-axis with rotation angle ϕ .

void rotate3DY (dmat33 &dst, const double phi)

Calculate the rotation matrix \mathbf{R} which represents the rotation along Y-axis with rotation angle ϕ .

void rotate3DZ (dmat33 &dst, const double phi)

Calculate the rotation matrix ${f R}$ which represents the rotation along Z-axis with rotation angle ϕ .

void alignZ (dmat33 &dst, const dvec3 &vec)

Calculate the rotation matrix ${\bf R}$ which aligns a direction vector ${\bf v}$ to Z-axis.

• void rotate3D (dmat33 &dst, const double phi, const dvec3 &axis)

Calculate the rotation matrix \mathbf{R} which represents the rotation along the axis \mathbf{v} with rotation angle ϕ .

void reflect3D (dmat33 &dst, const dvec3 &plane)

Calculate the transformation matrix ${\bf M}$ of reflection against a certian plane, which is represented by its normal vector ${\bf n}$.

void swingTwist (dvec4 &swing, dvec4 &twist, const dvec4 &src, const dvec3 &vec)

Calculate the two quaternions $\mathbf{q_s}$ and $\mathbf{q_t}$, which represent swing and twist along axis \mathbf{v} respectively, representing the rotation represented by quaternion \mathbf{q} .

void randRotate2D (dmat22 &rot)

Sample a 2D rotation matrix ${f R}$ from even distribution.

void randRotate3D (dmat33 &rot)

Sample a 3D rotation matrix ${f R}$ from even distribution.

4.1.1 Detailed Description

Euler.h contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution.

Author

Mingxu Hu Hongkun Yu

Version

1.4.11.080913

Copyright

THUNDER Non-Commercial Software License Agreement

ChangeLog

AUTHOR	TIME	VERSION	DESCRIPTION
Mingxu Hu	2015/03/23	0.0.1.050323	new file
Mingxu Hu	2018/09/13	1.4.11.080913	add more header

Quaternions are a number system that extends the complex numbers. Unit quaternions provide a convenient mathematical notation for representing rotations of objects in 3D. Compared to Euler angles, they are simpler to compose and aovid the problem of glimbal lock. Compared to rotation matrices, they are more compact and more efficient. Moroever, unlike Euler angles, unit quaternions do not rely on the choosing and order of the rotation axes.

To be noticed, Euler angles in this file follow the standard of ZXZ Euler system. In other words, Euler angle set $\{\phi,\theta,\psi\}$ stands for rotating along Z axis with ϕ , followed by rotating along X axis with θ , and followed by rotating along Z axis with ψ .

4.1.2 Function Documentation

4.1.2.1 alignZ()

Calculate the rotation matrix ${\bf R}$ which aligns a direction vector ${\bf v}$ to Z-axis.

Parameters

out	dst	\mathbf{R}
in	vec	v

4.1.2.2 angle() [1/3]

Calculate ϕ and θ given a certain direction ${\bf v}$.

v must be a unit vector. Output value ϕ ranges $[0, 2\pi)$, and θ ranges $[0, \pi]$.

out	phi	ϕ
out	theta	θ
in	src	\mathbf{v}

Calculate ϕ , θ and ψ of the rotation represented by the rotation matrix ${\bf R}.$

 ${f R}$ must be an orthogonal matirx and determinant of which equals to 1. In other words, $RR^T=I$ and $\det A=1$. Output value ϕ ranges $[0,2\pi)$, θ ranges $[0,\pi]$, and ψ ranges $[0,2\pi)$.

Parameters

out	phi	ϕ
out	theta	θ
out	psi	ψ
in	src	R

4.1.2.4 angle() [3/3]

Calculate ϕ , θ and ψ of the rotation represented by the unit quaternion \mathbf{q} .

Parameters

out	phi	ϕ
out	theta	θ
out	psi	ψ
in	src	q

4.1.2.5 direction()

Caclulate the unit direction vector ${f v}$, given the rotation and	gle ϕ and $ heta$.
,,0	

Parameters

out	dst	v
in	phi	ϕ
in	theta	θ

4.1.2.6 quaternion() [1/3]

Calculate the unit quaternion ${\bf q}$ for representing the rotation, given 3 Euler angles ϕ , θ and ψ .

Parameters

out	dst	q
in	phi	ϕ
in	theta	θ
in	psi	ψ

4.1.2.7 quaternion() [2/3]

Calculate the unit quaternion ${\bf q}$ for representing the rotation, given the rotation axis ${\bf r}$ and the rotation angle around this axis ϕ .

This rotation axis ${\bf r}$ must be a unit vector, while the rotation angle ϕ ranges $(-\infty, +\infty)$.

out	dst	q
in	phi	ϕ
in	axis	r

4.1.2.8 quaternion() [3/3]

Calculate the unit quaternion ${\bf q}$ for representing the rotation, given the rotation matrix ${\bf R}$.

Parameters

out	dst	\mathbf{q}
in	src	R

4.1.2.9 quaternion_conj()

Calculate the conjugate quaternion of a quaternion.

Returns

the conjugate quaternion

Parameters

in quat a quaternion	<u> </u>
----------------------	----------

4.1.2.10 quaternion_mul()

Calculate the product of two quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$.

Assuming that $\mathbf{q_1}=(w_1,x_1,y_1,z_1)$ and $\mathbf{q_2}=(w_2,x_2,y_2,z_2)$, the product can be calculated as

$$\begin{pmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{pmatrix} \times \begin{pmatrix} w_2 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2 \end{pmatrix}$$

Parameters

out	dst	product, a quaternion
in	а	left multiplier, $\mathbf{q_1}$
in	b	right multiplier, ${f q_2}$

4.1.2.11 randRotate2D()

Sample a 2D rotation matrix ${\bf R}$ from even distribution.

Parameters

```
out rot R
```

4.1.2.12 randRotate3D()

```
void randRotate3D ( {\tt dmat33~\&~rot~)}
```

Sample a 3D rotation matrix ${\bf R}$ from even distribution.

Parameters

```
out rot R
```

4.1.2.13 reflect3D()

```
void reflect3D ( \label{eq:dmat33 \& dst,}  const dvec3 & plane )
```

Calculate the transformation matrix ${\bf M}$ of reflection against a certian plane, which is represented by its normal vector ${\bf n}$.

out	dst	\mathbf{M}
in	plane	n

4.1.2.14 rotate2D() [1/2]

Calculate the rotation matrix (2D) \mathbf{R} , which rotates the unit vector $\mathbf{v_0} = \{1, 0\}$ to the given unit vector \mathbf{v} .

Parameters

out	dst	R
in	vec	v

4.1.2.15 rotate2D() [2/2]

Calculate the rotation matrix (2D) \mathbf{R} , given the rotation angle ϕ .

Parameters

out	dst	\mathbf{R}
in	phi	ϕ

4.1.2.16 rotate3D() [1/3]

Caclulate the rotation matrix ${\bf R},$ given the rotation angle $\phi,$ θ and $\psi.$

out	dst	\mathbf{R}
in	phi	ϕ
in	theta	θ
in	psi	ψ

4.1.2.17 rotate3D() [2/3]

```
void rotate3D ( \label{eq:dmat33 \& dst,}  const dvec4 & src )
```

Calculate the rotation matrix ${\bf R}$, given the unit quaternion ${\bf q}$ which represents this rotation.

Parameters

out	dst	R
in	src	\mathbf{q}

4.1.2.18 rotate3D() [3/3]

Calculate the rotation matrix \mathbf{R} which represents the rotation along the axis \mathbf{v} with rotation angle ϕ .

Parameters

out	dst	R
in	phi	ϕ
in	axis	v

4.1.2.19 rotate3DX()

```
void rotate3DX ( \label{eq:dmat33 \& dst,} $$ const double $phi$ )
```

Calculate the rotation matrix ${f R}$ which represents the rotation along X-axis with rotation angle ϕ .

out	dst	R
in	phi	ϕ

4.1.2.20 rotate3DY()

Calculate the rotation matrix ${f R}$ which represents the rotation along Y-axis with rotation angle ϕ .

Parameters

out	dst	\mathbf{R}
in	phi	ϕ

4.1.2.21 rotate3DZ()

```
void rotate3DZ ( \label{eq:dmat33 \& dst,}  const double phi )
```

Calculate the rotation matrix ${f R}$ which represents the rotation along Z-axis with rotation angle ϕ .

Parameters

out	dst	\mathbf{R}
in	phi	ϕ

4.1.2.22 swingTwist()

Calculate the two quaternions $\mathbf{q_s}$ and $\mathbf{q_t}$, which represent swing and twist along axis \mathbf{v} respectively, representing the rotation represented by quaternion \mathbf{q} .

out	swing	$\mathbf{q_s}$
out	twist	$\mathbf{q_t}$
in	src	\mathbf{q}
in	vec	v

4.2 include/Geometry/Transformation.h File Reference

Transformation.h contains several functions, for transformation of volume according to symmetry.

```
#include <cmath>
#include <iostream>
#include "Config.h"
#include "Macro.h"
#include "Typedef.h"
#include "Precision.h"
#include "Euler.h"
#include "Functions.h"
#include "Image.h"
#include "Volume.h"
#include "Symmetry.h"
```

Functions

void VOL_TRANSFORM_MAT_RL (Volume &dst, const Volume &src, const dmat33 &mat, const double r, const int interp)

Transform a volume in real space.

void VOL_TRANSFORM_MAT_FT (Volume &dst, const Volume &src, const dmat33 &mat, const double r, const int interp)

Transform a volume in Fourier space.

void SYMMETRIZE_RL (Volume &dst, const Volume &src, const Symmetry &sym, const double r, const int interp)

Transform a volume in real space according to symmetry.

 void SYMMETRIZE_FT (Volume &dst, const Volume &src, const Symmetry &sym, const double r, const int interp)

Transform a volume in Fourier space according to symmetry.

4.2.1 Detailed Description

Transformation.h contains several functions, for transformation of volume according to symmetry.

Author

Mingxu Hu

Version

1.4.11.080913

Copyright

THUNDER Non-Commercial Software License Agreement

ChangeLog

AUTHOR	TIME	VERSION	DESCRIPTION
Mingxu Hu	2015/03/23	0.0.1.050323	new file
Xiao Long	2018/09/13	1.4.11.080913	add documentation

For volumes in real space, the right transformation matrices, according to symmetry, set each voxel's value by interpolation in Fourier space. And vice versa for volumes in Fourier space.

4.2.2 Function Documentation

4.2.2.1 SYMMETRIZE_FT()

Transform a volume in Fourier space according to symmetry.

For volume dst in Fourier space(restricted to radius r in real space), generate the right transformation matrices according to symmetry sym for each symmetry elements. Then set the transformed volume dst by interp type interpolation.

Parameters

out	dst	the transformed volume in Fourier space
in	src	the original volume in Fourier space
in	sym	the volumes's symmetry, generates the right transformation matrices
in	r	the radius in real space, restricts the transformed volume's range
in	interp	the type of interpolation

4.2.2.2 SYMMETRIZE_RL()

Transform a volume in real space according to symmetry.

For volume dst in real space(restricted to radius r in Fourier space), generate the right transformation matrices according to symmetry sym for each symmetry elements. Then set the transformed volume dst by interp type interpolation.

Parameters

out	dst	the transformed volume in real space
in	src	the original volume in real space
in	sym	the volumes's symmetry, generates the right transformation matrices
in	r	the radius in Fourier space, restricts the transformed volume's range
in	interp	the type of interpolation

4.2.2.3 VOL_TRANSFORM_MAT_FT()

Transform a volume in Fourier space.

For each voxel of volume dst in Fourier space(restricted to radius r in real space), transform the Fourier space coordinate into the real space coordinate by transformation matrix mat. Then calculate the Fourier space value by interpolation(interp indicates the type of interpolation) according to its real space coordinate and set transformed volume src.

Parameters

out	dst	the transformed volume in Fourier space
in	src	the original volume in Fourier space
in	mat	the transformation matrix
in	r	the radius in real spcae, restricts the transformed volume's range
in	interp	the type of interpolation

4.2.2.4 VOL_TRANSFORM_MAT_RL()

Transform a volume in real space.

For each voxel of volume dst in real space(restricted to radius r in Fourier space), transform the real space coordinate into the Fourier space coordinate by transformation matrix mat. Then calculate the real space value by interpolation(interp indicates the type of interpolation) according to its Fourier space coordinate and set transformed volume src.

Parameters

out	dst	the transformed volume in real space
in	src	the original volume in real space
in	mat	the transformation matrix
in	r	the radius in Fourier spcae, restricts the transformed volume's range
in	interp	the type of interpolation

4.3 include/Image/MRCHeader.h File Reference

MRCHeader.h contains the main header of the MRC format, including fixed format values for metadata about the images/volumes.

Classes

struct MRCHeader

The constitution of MRC main header.

4.3.1 Detailed Description

MRCHeader.h contains the main header of the MRC format, including fixed format values for metadata about the images/volumes.

Author

Version

1.4.11.080913

Copyright

THUNDER Non-Commercial Software License Agreement

ChangeLog

AUTHOR	TIME	VERSION	DESCRIPTION
Mingxu Hu	2015/03/23	0.0.1.050323	new file
Xiao Long	2018/09/13	1.4.11.080913	add documentation

MRC is a file format that has become industry standard in cryo-electron microscopy (cryoEM) and electron tomography (ET), where the result of the technique is a three-dimensional grid of voxels each with a value corresponding to electron density or electric potential.

Reference: Cheng, Anchi; Henderson, Richard; Mastronarde, David; Ludtke, Steven J.; Schoenmakers, Remco H.M.; Short, Judith; Marabini, Roberto; Dallakyan, Sargis; Agard, David; Winn, Martyn (November 2015). "MR← C2014: Extensions to the MRC format header for electron cryo-microscopy and tomography". Journal of Structural Biology. 192 (2): 146–150. doi:10.1016/j.jsb.2015.04.002.

Index

alignZ	dmean, 6
Euler.h, 13	dmin, 6
angle	extra, 6
Euler.h, 13, 14	ispg, 6
	label, 7
cella	machst, 7
MRCHeader, 6	map, 7
cellb	mapc, 7
MRCHeader, 6	mapr, 7
	maps, 7
direction	mode, 7
Euler.h, 14	mx, 8
dmax	my, <mark>8</mark>
MRCHeader, 6	mz, 8
dmean	nlabels, 8
MRCHeader, 6	nsymbt, 8
dmin	nx, 8
MRCHeader, 6	nxstart, 8
	ny, 9
Euler.h	nystart, 9
alignZ, 13	nz, 9
angle, 13, 14	nzstart, 9
direction, 14	origin, 9
quaternion, 16	rms, 9
quaternion_conj, 17	machst
quaternion_mul, 17	
randRotate2D, 18	MRCHeader, 7
randRotate3D, 18	map
reflect3D, 18	MRCHeader, 7
rotate2D, 19	mapc
rotate3DX, 20	MRCHeader, 7
rotate3DY, 20	mapr
rotate3DZ, 21	MRCHeader, 7
rotate3D, 19, 20	maps
swingTwist, 21	MRCHeader, 7
extra	mode
MRCHeader, 6	MRCHeader, 7
	mx
include/Geometry/Euler.h, 11	MRCHeader, 8
include/Geometry/Transformation.h, 22	my
include/Image/MRCHeader.h, 25	MRCHeader, 8
ispg	mz
MRCHeader, 6	MRCHeader, 8
	olah ala
label	nlabels
MRCHeader, 7	MRCHeader, 8
MDOUL	nsymbt
MRCHeader, 5	MRCHeader, 8
cella, 6	nx
cellb, 6	MRCHeader, 8
dmax, 6	nxstart

28 INDEX

```
MRCHeader, 8
ny
    MRCHeader, 9
nystart
    MRCHeader, 9
nz
    MRCHeader, 9
nzstart
    MRCHeader, 9
origin
    MRCHeader, 9
quaternion
    Euler.h, 16
quaternion_conj
    Euler.h, 17
quaternion_mul
    Euler.h, 17
randRotate2D
    Euler.h, 18
randRotate3D
    Euler.h, 18
reflect3D
    Euler.h, 18
rms
    MRCHeader, 9
rotate2D
    Euler.h, 19
rotate3DX
    Euler.h, 20
rotate3DY
    Euler.h, 20
rotate3DZ
    Euler.h, 21
rotate3D
    Euler.h, 19, 20
SYMMETRIZE FT
    Transformation.h, 23
SYMMETRIZE_RL
    Transformation.h, 23
swingTwist
    Euler.h, 21
Transformation.h
    SYMMETRIZE_FT, 23
    SYMMETRIZE_RL, 23
    VOL_TRANSFORM_MAT_FT, 24
    VOL_TRANSFORM_MAT_RL, 24
VOL_TRANSFORM_MAT_FT
    Transformation.h, 24
VOL_TRANSFORM_MAT_RL
    Transformation.h, 24
```