THUNDER

Generated by Doxygen 1.8.13

Contents

Chapter 1

File Index

1.1 File List

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

Chapter 2

File Documentation

2.1 include/CTF.h File Reference

CTF.h contains several functions to satisfy the need of CTF calculations for various conditions, such as 1D, 2D and certain pixels. All these operations are carried in Fourier space with the variable spatial frequency.

```
#include "Complex.h"
#include "Functions.h"
#include "Image.h"
#include "Precision.h"
```

Functions

- RFLOAT CTF (const RFLOAT f, const RFLOAT voltage, const RFLOAT defocus, const RFLOAT CS, const RFLOAT amplitudeContrast, const RFLOAT phaseShift)

  This function returns the 1D CTF with paremeters given in Fourier space.
- void CTF (Image &dst, const RFLOAT pixelSize, const RFLOAT voltage, const RFLOAT defocusU, const R↩ FLOAT defocusV, const RFLOAT theta, const RFLOAT Cs, const RFLOAT amplitudeContrast, const RFLOAT phaseShift)

  This function is used to calculate the 2D CTF of a given image in Fourier space, output into the image I.
- void CTF (Image &dst, const RFLOAT pixelSize, const RFLOAT voltage, const RFLOAT defocusU, const R↩ FLOAT defocusV, const RFLOAT theta, const RFLOAT Cs, const RFLOAT amplitudeContrast, const RFLOAT phaseShift, const RFLOAT r)

  This function calculates CTF within a cut-off spatial frequency r in Fourier space, output into the image I.
- void CTF (RFLOAT ∗dst, const RFLOAT pixelSize, const RFLOAT voltage, const RFLOAT defocusU, const RFLOAT defocusV, const RFLOAT theta, const RFLOAT Cs, const RFLOAT amplitudeContrast, const RFL↩ OAT phaseShift, const int nCol, const int nRow, const int ∗iCol, const int ∗iRow, const int nPxl)

  This function provides a convenient way to compute certain pixel-positions' CTF values in Fourier space, output in a float array I.

2.1.1 Detailed Description

CTF.h contains several functions to satisfy the need of CTF calculations for various conditions, such as 1D, 2D and certain pixels. All these operations are carried in Fourier space with the variable spatial frequency.

Author

Mingxu Hu
Shouqing Li

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
Shouqing Li	2018/09/13	1.4.11.080913	add doc

Noted that, $CTF = -\sqrt{1-A^2}\sin(\chi) + A\cos(\chi)$ is the function used to calculate CTF, in which A is the fraction of total contrast attributed to amplitude contrast and χ denotes the function $\chi = \pi\lambda\Delta f H^2 + \frac{\pi}{2}C_S\lambda^3 H^4 - \Delta\varphi$, the effect of defocus Δf, spherical aberration C_S and an additional phase shift $\Delta\varphi$. H means the spatial frequency. When it comes to the 2D condition, the astigmatism should also be taken into consideration through $\Delta f = \frac{1}{2}[\Delta f_1 + \Delta f_2 + \Delta\Delta f\cos(2[\alpha_g - \alpha_{ast}])]$, where Δf_1 and Δf_2 are the len's defocus along normal directions, $\Delta\Delta f = \Delta f_1 - \Delta f_2$, α_g represents the angle between vector of pixel choosed and X axis and α_{ast} is the angle between X axis and Δf_1 direction. V in V. H in Å^{-1}. r in Å^{-1}. Δf in Å. Δf_1 in Å. Δf_2 in Å. λ in Å. a in Å. $\Delta\varphi$ in rad. α_g in rad. α_{ast} in rad.

2.1.2 Function Documentation

2.1.2.1 CTF() [1/4]

```
RFLOAT CTF (
            const RFLOAT f,
            const RFLOAT voltage,
            const RFLOAT defocus,
            const RFLOAT CS,
            const RFLOAT amplitudeContrast,
            const RFLOAT phaseShift )
```

This function returns the 1D CTF with paremeters given in Fourier space.

The input of voltage determines the value of wavelenghth λ. The other variables are substituded into CTF equation computing 1D CTF.

Parameters

in	*f*	H
in	*voltage*	V
in	*defocus*	Δf
in	*CS*	C_S
in	*amplitudeContrast*	A
in	*phaseShift*	$\Delta\varphi$

2.1.2.2 CTF() [2/4]

```
void CTF (
            Image & dst,
            const RFLOAT pixelSize,
            const RFLOAT voltage,
            const RFLOAT defocusU,
            const RFLOAT defocusV,
            const RFLOAT theta,
            const RFLOAT Cs,
            const RFLOAT amplitudeContrast,
            const RFLOAT phaseShift )
```

This function is used to calculate the 2D CTF of a given image in Fourier space, output into the image I.

Assign the computed 2D CTF values to every image pixel after getting the column and row number of it.

Parameters

in,out	*dst*	I
in	*pixelSize*	a
in	*voltage*	V
in	*defocusU*	Δf_1
in	*defocusV*	Δf_2
in	*theta*	$\alpha_{\alpha st}$
in	*Cs*	C_S
in	*amplitudeContrast*	A
in	*phaseShift*	$\Delta\varphi$

2.1.2.3 CTF() [3/4]

```
void CTF (
            Image & dst,
            const RFLOAT pixelSize,
            const RFLOAT voltage,
            const RFLOAT defocusU,
```

```
              const RFLOAT defocusV,
              const RFLOAT theta,
              const RFLOAT Cs,
              const RFLOAT amplitudeContrast,
              const RFLOAT phaseShift,
              const RFLOAT r )
```

This function calculates CTF within a cut-off spatial frequency r in Fourier space, output into the image I.

Parameters

in,out	*dst*	I
in	*pixelSize*	a
in	*voltage*	V
in	*defocusU*	Δf_1
in	*defocusV*	Δf_2
in	*theta*	$\alpha_{\alpha st}$
in	*Cs*	C_S
in	*amplitudeContrast*	A
in	*phaseShift*	$\Delta\varphi$
in	*r*	r

2.1.2.4 CTF() [4/4]

```
void CTF (
              RFLOAT * dst,
              const RFLOAT pixelSize,
              const RFLOAT voltage,
              const RFLOAT defocusU,
              const RFLOAT defocusV,
              const RFLOAT theta,
              const RFLOAT Cs,
              const RFLOAT amplitudeContrast,
              const RFLOAT phaseShift,
              const int nCol,
              const int nRow,
              const int * iCol,
              const int * iRow,
              const int nPxl )
```

This function provides a convenient way to compute certain pixel-positions' CTF values in Fourier space, output in a float array I.

X and Y are the number of columns and rows of the given image in real space. x and y mean the column and row number of a certain pixel. N represents the total number of pixels to be calculated. All the results will be assigned to these points respectively. The spatial frequency H can be got by formula $H = \sqrt{(\frac{x_i}{aX})^2 + (\frac{y_i}{aY})^2}$, the range of i is from 0 to N-1.

Parameters

out	*dst*	I

Parameters

in	*pixelSize*	a
in	*voltage*	V
in	*defocusU*	Δf_1
in	*defocusV*	Δf_2
in	*theta*	$\alpha_{\alpha st}$
in	*Cs*	C_S
in	*amplitudeContrast*	A
in	*phaseShift*	$\Delta\varphi$
in	*nCol*	X
in	*nRow*	Y
in	*iCol*	x_i
in	*iRow*	y_i
in	*nPxl*	N

2.2 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 $\mathbf{q_1}$ *and* $\mathbf{q_2}$*.*
- 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* $\mathbf{R}$*.*
- 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 $\mathbf{q}$ for representing the rotation, given the rotation matrix $\mathbf{R}$.
- 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 $\mathbf{R}$ which represents the rotation along Z-axis with rotation angle ϕ.
- void alignZ (dmat33 &dst, const dvec3 &vec)

  Calculate the rotation matrix $\mathbf{R}$ which aligns a direction vector $\mathbf{v}$ 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 $\mathbf{M}$ of reflection against a certian plane, which is represented by its normal vector $\mathbf{n}$.
- 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 $\mathbf{R}$ from even distribution.
- void randRotate3D (dmat33 &rot)

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

2.2.1 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

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.

2.2.2 Function Documentation

2.2.2.1 alignZ()

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

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

Parameters

out	*dst*	$\mathbf{R}$
in	*vec*	$\mathbf{v}$

2.2.2.2 angle() [1/3]

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

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

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

Parameters

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

2.2.2.3 angle() [2/3]

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

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

$\mathbf{R}$ 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*	$\mathbf{R}$

2.2.2.4 angle() [3/3]

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

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

Parameters

out	*phi*	ϕ
out	*theta*	θ
out	*psi*	ψ
in	*src*	$\mathbf{q}$

2.2.2.5 direction()

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

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

Parameters

out	*dst*	$\mathbf{v}$
in	*phi*	ϕ
in	*theta*	θ

2.2.2.6 quaternion() [1/3]

```
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.

Parameters

out	*dst*	$\mathbf{q}$
in	*phi*	ϕ
in	*theta*	θ
in	*psi*	ψ

2.2.2.7 quaternion() [2/3]

```
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.

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

Parameters

out	*dst*	$\mathbf{q}$
in	*phi*	ϕ
in	*axis*	$\mathbf{r}$

2.2.2.8 quaternion() [3/3]

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

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

Parameters

out	*dst*	$\mathbf{q}$
in	*src*	$\mathbf{R}$

2.2.2.9 quaternion_conj()

```
dvec4 quaternion_conj (
            const dvec4 & quat )
```

Calculate the conjugate quaternion of a quaternion.

Returns

the conjugate quaternion

Parameters

in	*quat*	a quaternion

2.2.2.10 quaternion_mul()

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

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

$$\begin{pmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{pmatrix} \times \begin{pmatrix} w_2 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2 \\ w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2 \\ w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2 \end{pmatrix}$$

Parameters

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

2.2.2.11 randRotate2D()

```
void randRotate2D (
            dmat22 & rot )
```

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

Parameters

out	*rot*	$\mathbf{R}$

2.2.2.12 randRotate3D()

```
void randRotate3D (
            dmat33 & rot )
```

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

Parameters

out	*rot*	$\mathbf{R}$

2.2.2.13 reflect3D()

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

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

Parameters

out	*dst*	$\mathbf{M}$
in	*plane*	$\mathbf{n}$

2.2.2.14 rotate2D() [1/2]

```
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 $\mathbf{v}$.

Parameters

out	*dst*	$\mathbf{R}$
in	*vec*	$\mathbf{v}$

2.2.2.15 rotate2D() [2/2]

```
void rotate2D (
            dmat22 & dst,
            const double phi )
```

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

Parameters

out	*dst*	$\mathbf{R}$
in	*phi*	ϕ

2.2.2.16 rotate3D() [1/3]

```
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.

Parameters

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

2.2.2.17 rotate3D() [2/3]

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

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

Parameters

out	*dst*	$\mathbf{R}$
in	*src*	$\mathbf{q}$

2.2.2.18 rotate3D() [3/3]

```
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.

Parameters

out	*dst*	$\mathbf{R}$
in	*phi*	ϕ
in	*axis*	$\mathbf{v}$

2.2.2.19 rotate3DX()

```
void rotate3DX (
            dmat33 & dst,
            const double phi )
```

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

Parameters

out	*dst*	$\mathbf{R}$
in	*phi*	ϕ

2.2.2.20 rotate3DY()

```
void rotate3DY (
            dmat33 & dst,
            const double phi )
```

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

Parameters

out	*dst*	$\mathbf{R}$
in	*phi*	ϕ

2.2.2.21 rotate3DZ()

```
void rotate3DZ (
            dmat33 & dst,
            const double phi )
```

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

Parameters

out	*dst*	$\mathbf{R}$
in	*phi*	ϕ

2.2.2.22 swingTwist()

```
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 $\mathbf{q}$.

Parameters

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

Index