２眼ステレオ３次元計測ライブラリ

(Ver.2.3)

2018 年 8 月

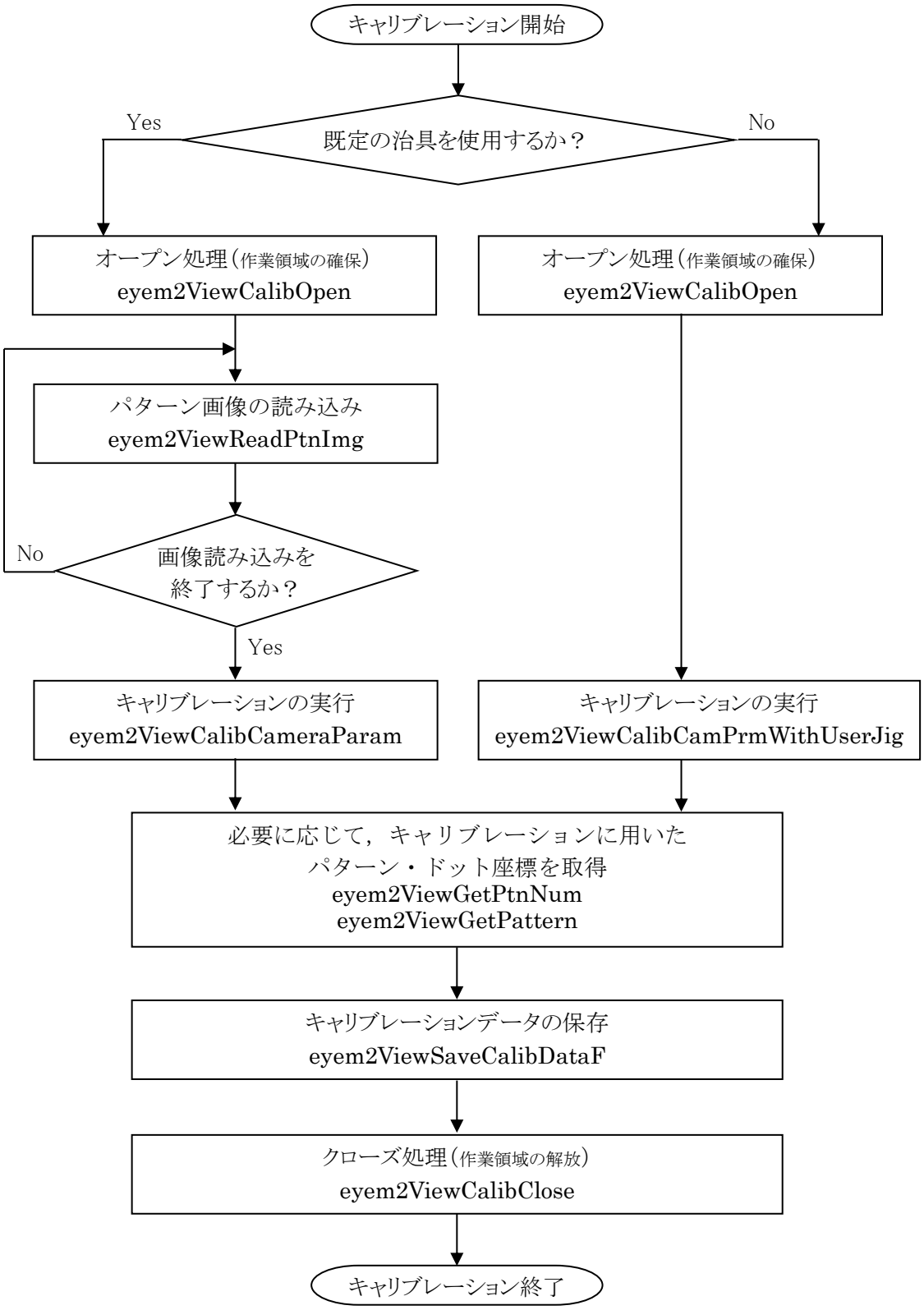
株式会社 アイディール

目次

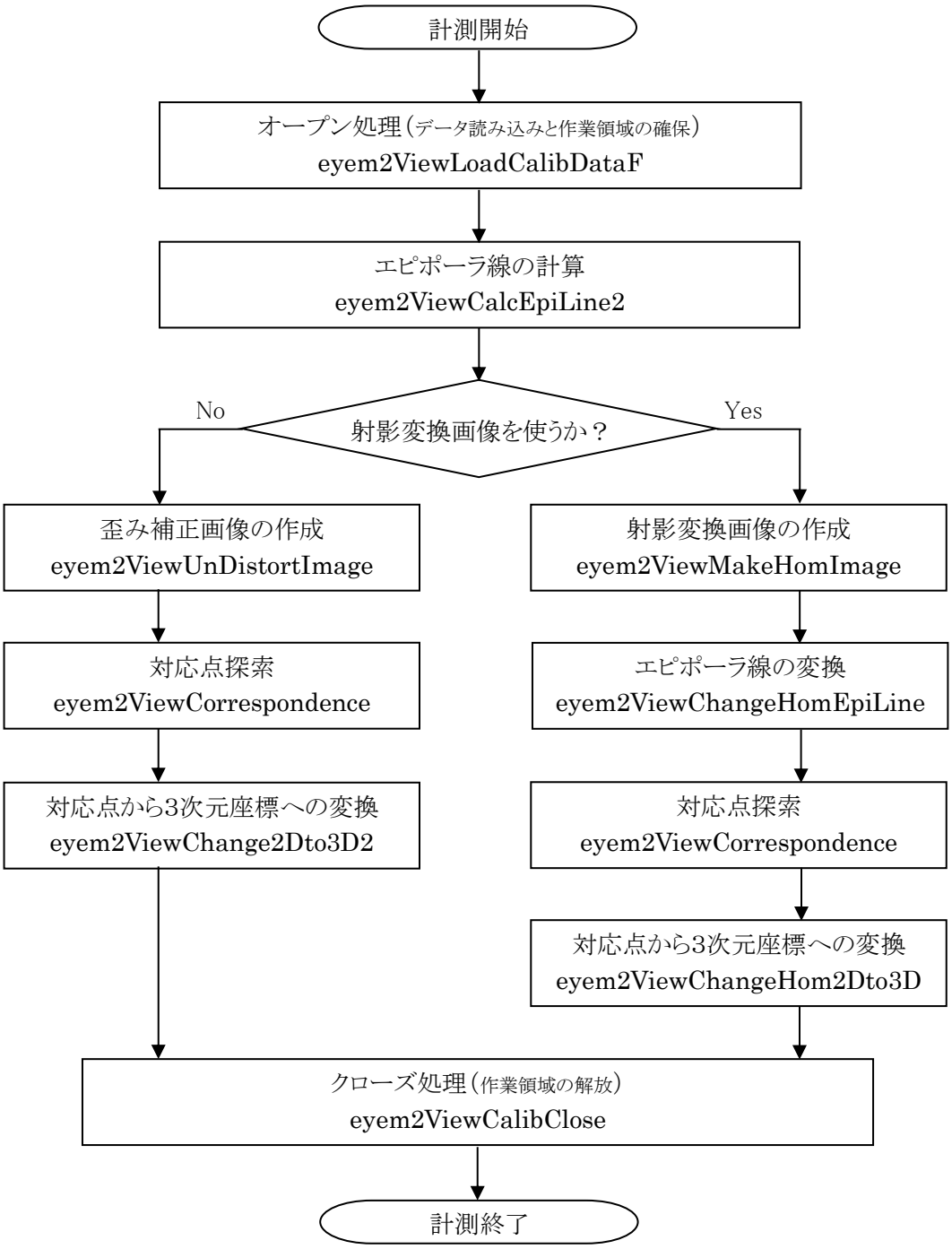
1．2眼ステレオ3次元計測ライブラリの使用手順

2眼ステレオ3次元計測ライブラリでは，キャリブレーションフェーズと計測フェーズがあります．これらの使用手順は以下の流れとなります．

（1） キャリブレーションフェーズ



（2） 計測フェーズ



2. ライブラリの詳細説明

次ページ以降に各々のライブラリの説明を行います．なお，ライブラリ内に出てくる定数および構造体の具体的内容は **Appendix1** に示しましたので，そちらをご参照ください．

eyem2ViewCalibOpen

機 能　キャリブレーションのオープン処理

形 式

```
#include "eyem2View.h"
int    eyem2ViewCalibOpen( int iSetNum, Eyem2ViewJig *tpJig, Eyem2ViewCam *tpCam,
                           int iWldFlag, void **vpCalib );
```

解 説　作業領域の確保を行い，キャリブレーション・ディスクリプタを生成します．

引 数

iSetNum　［入力］キャリブレーション・パターンの左右カメラ画像セットの数．
３セット以上で指定します．

*tpJig　［入力］既定のキャリブレーション治具情報．
既定以外の治具を用いる場合は，NULLを指定してください．なお，既定治具の詳細は **Appendix2** をご参照ください．

*tpCam　［入力］カメラ情報．
iCamL：左カメラ種別．以下のいずれかを指定します．
iCamR：右カメラ種別．以下のいずれかを指定します．

EYEM_2VIEW _CAM_NORMAL	通常のカメラ
EYEM_2VIEW _CAM_SCHEIMPFLUG	シャインプルーフ構成のカメラ
EYEM_2VIEW_CAM_TELECENTRIC	テレセントリックレンズ使用のカメラ

iWidthL：左カメラの画像メモリX方向サイズ（画素）．
iWidthR：右カメラの画像メモリX方向サイズ（画素）．
iHeightL：左カメラの画像メモリY方向サイズ（画素）．
iHeightR：右カメラの画像メモリY方向サイズ（画素）．
（注1）テレセントリックレンズ使用カメラを含む場合は，必ずその1つを「左カメラ」としてください．
（注2）カメラが2台ともテレセントリックレンズ使用の場合は，キャリブレーションはできますが，3次元計測はできません．

iWldFlag　［入力］ワールド座標系の種別．
以下のいずれかを指定します．なお，座標系の詳細は **Appendix3** をご参照ください．

EYEM_2VIEW_WLD_PTN	第1パターン座標系
EYEM_2VIEW_WLD_CAM_LEFT	左カメラ座標系
EYEM_2VIEW_WLD_CAM_RIGHT	右カメラ座標系
EYEM_2VIEW_WLD_CAM_MIDDLE	中間点座標系

（注）テレセントリックレンズを使用の場合は，第1パターン座標系に限り選択可能となります．

**vpCalib　　［出力］キャリブレーション・ディスクリプタ（データ領域の先頭アドレス）．
なお，*vpCalib は 必ず NULL に初期設定してください．

戻り値　　終了コードです．

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	ワークメモリ不足
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　　本ライブラリを使用した際は，キャリブレーション作業終了時に eyem2ViewCalibClose関数を必ず実行してください．さもないと，作業領域が解放されません．

eyem2ViewSaveCalibDataF

機 能　キャリブレーションデータのファイル保存

形 式

```
#include “eyem2View.h”
int    eyem2ViewSaveCalibDataF( const char *cpFilePath, void *vpCalib );
```

解 説　キャリブレーションデータを指定されたファイルに保存します.

引 数

*cpFilePath　［入力］保存先のファイル・パス.

*vpCalib　［入力］キャリブレーション・ディスクリプタ.
eyem2ViewCalibOpen関数の出力値を指定します.

戻り値　終了コードです.

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可(ファイル保存失敗)
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません.

eyem2ViewLoadCalibDataF

機 能　計測のオープン処理

形 式

```
#include “eyem2View.h”
int    eyem2ViewLoadCalibDataF( const char *cpFilePath, void **vpCalib );
```

解 説　保存してあるファイルからキャリブレーションデータを読み込み，キャリブレーション・ディスクリプタを生成します．また，作業領域の確保を行います．

引 数

*cpFilePath　［入力］保存してあるキャリブレーションデータのファイル・パス．

**vpCalib　［出力］キャリブレーション・ディスクリプタ（データ領域の先頭アドレス）．
なお，*vpCalib は 必ず NULL に初期設定してください．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	ワークメモリ不足
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可（ファイル読み込み失敗）
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　本ライブラリを使用した際は，計測終了時に eyem2ViewCalibClose関数 を必ず実行してください．さもないと，作業領域が解放されません．

eyem2ViewCalibClose

機 能　クローズ処理

形 式

```
#include “eyem2View.h”
void　eyem2ViewCalibClose( void **vpCalib );
```

解 説　キャリブレーションまたは計測で使用したワークメモリの解放を行います.

引 数

**vpCalib　［入・出力］キャリブレーション・ディスクリプタ.
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します. *vpCalib は NULL に設定して戻します.

戻り値　ありません.

留意事項　特にありません.

eyem2ViewReadPtnImg

機 能　左右カメラのパターン・ドット座標点の読み込み（既定治具）

形 式

```
#include "eyem2View.h"
int    eyem2ViewReadPtnImg( void *vpCalib, int iImgNo, Eyem2ViewImg *tpImgSet,
                            EyemRect *tpRoiL, EyemRect *tpRoiR );
```

解 説　既定治具を用いた左右カメラのパターン・ドット画像から，有効な座標点を読み込みます．

引 数

*vpCalib　［入・出力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数の出力値を指定します．

iImgNo　［入力］対象となる左右カメラのパターン画像セットの番号．
0番からの通し番号を入力します．

*tpImgSet　［入力］対象となる左右カメラのパターン画像セット．
なお，これらの画像の作成（キャリブレーションの方法）については，
Appendix4 をご参照ください．
*ucpImgCamL：左カメラ画像メモリの先頭アドレス．
*ucpImgCamR：右カメラ画像メモリの先頭アドレス．
（注）テレセントリックレンズ使用の場合は，（左カメラの）0番パターン画像のみが使用されます．よって，1番以降の ucpImgCamL には NULL を指定ください．

*tpRoiL　［入力］左カメラ用・処理範囲の矩形領域（左上座標およびサイズ）．
NULL を指定すると自動設定となります．
iX：左上X座標（画素）．
iY：左上Y座標（画素）．
iWidth：領域のX方向サイズ（画素）．
iHeight：領域のY方向サイズ（画素）．

*tpRoiR　［入力］右カメラ用・処理範囲の矩形領域（左上座標およびサイズ）．
NULL を指定すると自動設定となります．
iX：左上X座標（画素）．
iY：左上Y座標（画素）．
iWidth：領域のX方向サイズ（画素）．
iHeight：領域のY方向サイズ（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	ワークメモリ不足

FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_CANNOT_FIND_PLATE	プレートが見つからない
FUNC_FEW_PTN_SMPL_NUM	パターンのドット点数が足りない
FUNC_CANNOT_READ_MARK	パターン大ドット点が読めない
FUNC_CANNOT_READ_PTN	パターン・ドット点の実座標が読めない

留意事項　本関数は，eyem2ViewCalibOpen関数（治具情報指定）でのオープン時に限り使用可能となります．

eyem2ViewCalibCameraParam

機 能　カメラキャリブレーションの実行（既定治具）

形 式

```
#include "eyem2View.h"
int    eyem2ViewCalibCameraParam( void *vpCalib, Eyem2ViewBndl *tpBundle );
```

解 説　既定治具を用いた左右パターン画像（3セット以上）から，それぞれのカメラのキャリブレーションを個別に行います．

引 数

*vpCalib　［入・出力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数の出力値を指定します．

*tpBundle　［出力］バンドル調整結果（反復回数，残差RMS）．
iNumL：左カメラのバンドル調整反復回数．
dRmsL：左カメラの残差（再投影誤差）の重み付き平均二乗誤差の平方根（画素）．
iNumL：右カメラのバンドル調整反復回数．
dRmsL：右カメラの残差（再投影誤差）の重み付き平均二乗誤差の平方根（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	ワークメモリ不足
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_HOMOGRAPHY	ホモグラフィー行列計算失敗
FUNC_FAILED_CAM_PRM	カメラパラメータ計算失敗
FUNC_FAILED_BUNDLE_ADJ	バンドル調整失敗
FUNC_FAILED_FUNDAMENTAL	基礎行列計算失敗

留意事項　本関数は，eyem2ViewCalibOpen関数（治具情報指定）でのオープン時に限り使用可能となります．

eyem2ViewCalibCamPrmWithUserJig

機 能　カメラキャリブレーションの実行（既定以外の治具）

形 式

```
#include "eyem2View.h"
Int    eyem2ViewCalibCamPrmWithUserJig( void *vpCalib, EyemCalibPtn taPtnL[],
                                        EyemCalibPtn taPtnR[], Eyem2ViewBndl *tpBundle );
```

解 説　既定以外の治具を用いた左右パターン画像（3セット以上）の参照点（4点以上）により，それぞれのカメラのキャリブレーションを個別に行います．

引 数

*vpCalib　［入・出力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数の出力値を指定します．

taPtnL[]　［入力］左カメラのパターン参照点配列（iSetNum個の配列）
eyem2ViewCalibOpen関数で指定した画像セット数（iSetNum）分の配列です．なお，各パターンの参照点は，4点以上を用意してください．
iPtnRefN：パターン参照点の個数（4点以上）．
*tpObjPt：参照点の実座標（iPtnRefN個の配列，単位：mm）．
*tpImgPt：参照点の画像座標（iPtnRefN個の配列，単位：画素）．
（注）テレセントリックレンズ使用の場合は，インデックス[0]のパターン参照点のみが使用されます．よって，本配列の要素数は1で構いません．

taPtnR[]　［入力］右カメラのパターン参照点配列（iSetNum個の配列）
eyem2ViewCalibOpen関数で指定した画像セット数（iSetNum）分の配列です．なお，各パターンの参照点は，4点以上を用意してください．
iPtnRefN：パターン参照点の個数（4点以上）．
*tpObjPt：参照点の実座標（iPtnRefN個の配列，単位：mm）．
*tpImgPt：参照点の画像座標（iPtnRefN個の配列，単位：画素）．

*tpBundle　［出力］バンドル調整結果（反復回数，残差RMS）．
iNumL：左カメラのバンドル調整反復回数．
dRmsL：左カメラの残差（再投影誤差）の重み付き平均二乗誤差の平方根（画素）．
iNumL：右カメラのバンドル調整反復回数．
dRmsL：右カメラの残差（再投影誤差）の重み付き平均二乗誤差の平方根（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了

FUNC_NOT_ENOUGH_MEM	ワークメモリ不足
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FEW_PTN_SMPL_NUM	パターンの参照点数が足りない
FUNC_FAILED_HOMOGRAPHY	ホモグラフィー行列計算失敗
FUNC_FAILED_CAM_PRM	カメラパラメータ計算失敗
FUNC_FAILED_BUNDLE_ADJ	バンドル調整失敗
FUNC_FAILED_FUNDAMENTAL	基礎行列計算失敗

留意事項　本関数は，eyem2ViewCalibOpen関数（治具引数NULL指定）でのオープン時に限り使用可能となります．

eyem2ViewChange2Dto3D

機 能　　左右カメラの原画像の対応点から3次元座標点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewChange2Dto3D( void *vpCalib, EyemOcsDXY *tpImgPtL,
                              EyemOcsDXY *tpImgPtR, EyemOcsDXYZ *tpPt3D );
```

解 説　　左右カメラの原画像における対応点から，その3次元座標を求めます．

引 数

*vpCalib　　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

*tpImgPtL　　［入力］左カメラの原画像における座標．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpImgPtR　　［入力］右カメラの原画像における座標．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpPt3D　　［出力］3次元座標．
dX： X座標（mm）．
dY： Y座標（mm）．
dZ： Z座標（mm）．

戻り値　　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項

（1）2台ともテレセントリックレンズ使用カメラの場合は，使用できません（計算不可）．

（2）射影変換画像（eyem2ViewMakeHomImage関数の出力画像）上の対応点は，必ず原画像上の座標点に変換してから使用してください．この変換は，eyem2ViewInvHomImgPoint関数 で行います．

（3）歪み補正画像（eyem2ViewUnDistortImage関数の出力画像）上の対応点は，必ず原画像上の座標に変換してから使用してください．この変換は，eyem2ViewDistortPoint関数 で行います．

eyem2ViewChange2Dto3D2

機 能　左右カメラ画像の対応点から3次元座標点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewChange2Dto3D2( void *vpCalib, int iImgL, EyemOcsDXY *tpImgPtL,
                                int iImgR, EyemOcsDXY *tpImgPtR, EyemOcsDXYZ *tpPt3D );
```

解 説　左右カメラの指定画像における対応点から，その3次元座標を求めます．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

iImgL　［入力］左カメラの画像種別．
以下のいずれかを指定します．

EYEM_2VIEW_IMG_ACTUAL	原画像（レンズ歪みのある画像）
EYEM_2VIEW_IMG_IDEAL	理想画像（レンズ歪み補正画像）

*tpImgPtL　［入力］左カメラ画像における座標．
dX： X座標（画素）．
dY： Y座標（画素）．

iImgR　［入力］右カメラの画像種別．
以下のいずれかを指定します．

EYEM_2VIEW_IMG_ACTUAL	原画像（レンズ歪みのある画像）
EYEM_2VIEW_IMG_IDEAL	理想画像（レンズ歪み補正画像）

*tpImgPtR　［入力］右カメラ画像における座標．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpPt3D　［出力］3次元座標．
dX： X座標（mm）．
dY： Y座標（mm）．
dZ： Z座標（mm）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項

（1）２台ともテレセントリックレンズ使用カメラの場合は，使用できません（計算不可）．

（2）一方のカメラ画像が，射影変換画像（eyem2ViewMakeHomImage関数の出力画像）の場合は，eyem2ViewChangeHom2Dto3D関数を使用してください．

eyem2ViewChange3Dto2D

機 能　　3次元座標点から画像座標点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewChange3Dto2D( void *vpCalib, char cLR, EyemOcsDXYZ *tpPt3D,
                              EyemOcsDXY *tpPt2D );
```

解 説　　指定された3次元座標点を，指定カメラの画像面上へ投影したときの座標を求めます．

引 数

*vpCalib　　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　　［入力］投影する画像面．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*tpPt3D　　［入力］3次元座標．
dX： X座標（mm）．
dY： Y座標（mm）．
dZ： Z座標（mm）．

*tpPt2D　　［出力］画像座標．
dX： X座標（画素）．
dY： Y座標（画素）．

戻り値　　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　　特にありません．

eyem2ViewCalcEpiLine

機 能　エピポーラ線の計算

形 式

```
#include "eyem2View.h"
int    eyem2ViewCalcEpiLine( void *vpCalib, char cLR, EyemOcsDXY *tpImgPt,
                                    Eyem2ViewEpi *tpEpiLine );
```

解 説　左(または右)カメラの原画像上の指定点に対する，右(または左)カメラ画像上のエピポーラ線を求めます(正確には，歪み補正画像上のエピポーラ線が求まります)．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］エピポーラ線の乗る画像．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*tpImgPt　［入力］原画像における指定点の座標．
dX： X座標(mm)．
dY： Y座標(mm)．

*tpEpiLine　［出力］エピポーラ線情報(直線の係数，画像内線分端点)．
tLine：エピポーラ線($ax + by + c = 0$)の係数a, b, c．
taPt[2]：エピポーラ線と画像境界との交点(2点)の座標．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項　特にありません．

eyem2ViewCalcEpiLine2

機 能　エピポーラ線の計算（汎用）

形 式

```
#include "eyem2View.h"
int    eyem2ViewCalcEpiLine2( void *vpCalib, char cLR, int iImg, EyemOcsDXY *tpImgPt,
                              Eyem2ViewEpi *tpEpiLine );
```

解 説　左（または右）カメラ画像上の指定点に対する，右（または左）カメラ画像上のエピポーラ線を求めます（正確には，歪み補正画像上のエピポーラ線が求まります）．
なお，eyem2ViewCalcEpiLine 関数との違いは，指定点の乗る画像種別の入力があることです．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］エピポーラ線の乗る画像．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

iImg　［入力］指定点の乗る画像種別．
以下のいずれかを指定します．

EYEM_2VIEW_IMG_ACTUAL	原画像（レンズ歪みのある画像）
EYEM_2VIEW_IMG_IDEAL	理想画像（レンズ歪み補正画像）

*tpImgPt　［入力］指定点の座標．
dX： X座標（mm）．
dY： Y座標（mm）．

*tpEpiLine　［出力］エピポーラ線情報（直線の係数，画像内線分端点）．
tLine：エピポーラ線（$ax + by + c = 0$）の係数a, b, c．
taPt[2]：エピポーラ線と画像境界との交点（2点）の座標．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項　特にありません．

eyem2ViewChangeHomEpiLine

機 能　射影変換画像上のエピポーラ線への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewChangeHomEpiLine( void *vpCalib, char cLR, Eyem2ViewHom *tpParam,
                                  Eyem2ViewEpi *tpLineSrc, Eyem2ViewEpi *tpLineDst );
```

解 説　指定カメラの原画像上（正確には歪み補正画像上）のエピポーラ線（eyem2ViewCalcEpiLine 関数の出力値）を射影変換画像（eyem2ViewMakeHomImage 関数の出力画像）上のエピポーラ線に変換します.

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ.
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します.

cLR　［入力］エピポーラ線の乗る画像.
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します.

*tpParam　［入力］画像射影変換パラメータ.
必ず, eyem2ViewMakeHomImage関数で用いた値を指定します. なお, 画像射影変換については, **Appendix5** をご参照ください.
tCenter: 射影変換画像の中央に対応する, 原画像の座標点.
dWorldZ: 計測点3次元座標のz座標概略値.

*tpLineSrc　［入力］原画像上のエピポーラ線情報.
eyem2ViewCalcEpiLine関数の出力値を指定します.
tLine: エピポーラ線（$ax + by + c = 0$）の係数a, b, c.
taPt[2]: エピポーラ線と画像境界との交点（2点）の座標.

*tpLineDst　［出力］射影変換画像上のエピポーラ線情報.
tLine: エピポーラ線（$ax + by + c = 0$）の係数a, b, c.
taPt[2]: エピポーラ線と画像境界との交点（2点）の座標.

戻り値　終了コードです.

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません.

eyem2ViewCorrespondence

機 能　エピポーラ線上の対応点探索

形 式

```
#include "eyem2View.h"
int    eyem2ViewCorrespondence( void *vpCalib, int iMethod, char cLR,
                unsigned char *ucpRefImg, EyemRect *tpTmpRect, EyemOcsDXY *tpMeasPt,
                unsigned char *ucpObjImg, EyemRect *tpObjRect, Eyem2ViewEpi *tpEpiLine,
                EyemOcsDXY *tpCrspnd );
```

解 説　一方のカメラ画像で指定された計測点に対する，もう一方のカメラ画像上のエピポーラ線上で，対応点探索（サーチ）を行います．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

iMethod　［入力］サーチ方法．
以下のいずれかを指定します．

EYEM_2VIEW_SEARCH_ZNCC	ゼロ平均正規化相互相関法
EYEM_2VIEW_SEARCH_SAD	差分絶対総和法

cLR　［入力］探索の対象となる画像．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*ucpRefImg　［入力］サーチ・テンプレート用画像メモリの先頭アドレス．
eyem2ViewCalibOpen関数で指定した画像サイズとします．なお，テンプレート画像は，厳密には歪み補正画像を用いますが，精度上問題がなければ原画像でも構いません．

*tpTmpRect　［入力］サーチ・テンプレートとする矩形領域（左上座標およびサイズ）．
iX： 左上X座標（画素）．
iY： 左上Y座標（画素）．
iWidth： 領域のX方向サイズ（画素）．
iHeight： 領域のY方向サイズ（画素）．

*tpMeasPt　［入力］テンプレート領域内の計測対象点．
dX： X座標（画素）．
dY： Y座標（画素）．

*ucpObjImg　［入力］探索対象画像メモリの先頭アドレス．
eyem2ViewCalibOpen関数で指定した画像サイズとします．なお，探索対象画像は，厳密には歪み補正画像を用いますが，精度上問題がなければ原画像でも構いません．また，探索対象画像は，射影変換画像で

も構いません（射影変換画像は，歪み補正が施されています）．

*tpObjRect　［入力］探索範囲の矩形領域（左上座標およびサイズ）．

iX：左上X座標（画素）．

iY：左上Y座標（画素）．

iWidth：領域のX方向サイズ（画素）．

iHeight：領域のY方向サイズ（画素）．

*tpEpiLine　［入力］探索対象画像内のエピポーラ線情報（直線の係数のみ使用）．

eyem2ViewCalcEpiLine関数 または eyem2ViewChangeHomEpiLine関数の出力値を指定します．

tLine：エピポーラ線（$ax + by + c = 0$）の係数a, b, c．

taPt[2]：エピポーラ線と画像境界との交点（2点）の座標．

*tpCrspnd　［出力］探索対象画像内の対応点．

dX：X座標（画素）．

dY：Y座標（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	ワークメモリ不足
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません．

eyem2ViewCorrespondence2

機 能　エピポーラ線上の対応点探索（相関値出力）

形 式

```
#include "eyem2View.h"
int    eyem2ViewCorrespondence2( void *vpCalib, int iMethod, char cLR,
                unsigned char *ucpRefImg, EyemRect *tpTmpRect, EyemOcsDXY *tpMeasPt,
                unsigned char *ucpObjImg, EyemRect *tpObjRect, Eyem2ViewEpi *tpEpiLine,
                EyemOcsDXY *tpCrspnd, double *dpCorr );
```

解 説　一方のカメラ画像で指定された計測点に対する，もう一方のカメラ画像上のエピポーラ線上で，対応点探索（サーチ）を行います．
なお，eyem2ViewCorrespondence 関数との違いは，サーチ結果の相関値出力があることだけです．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

iMethod　［入力］サーチ方法．
以下のいずれかを指定します．

EYEM_2VIEW_SEARCH_ZNCC	ゼロ平均正規化相互相関法
EYEM_2VIEW_SEARCH_SAD	差分絶対総和法

cLR　［入力］探索の対象となる画像．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*ucpRefImg　［入力］サーチ・テンプレート用画像メモリの先頭アドレス．
eyem2ViewCalibOpen関数で指定した画像サイズとします．なお，テンプレート画像は，厳密には歪み補正画像を用いますが，精度上問題がなければ原画像でも構いません．

*tpTmpRect　［入力］サーチ・テンプレートとする矩形領域（左上座標およびサイズ）．
iX： 左上X座標（画素）．
iY： 左上Y座標（画素）．
iWidth： 領域のX方向サイズ（画素）．
iHeight： 領域のY方向サイズ（画素）．

*tpMeasPt　［入力］テンプレート領域内の計測対象点．
dX： X座標（画素）．
dY： Y座標（画素）．

*ucpObjImg　［入力］探索対象画像メモリの先頭アドレス．
eyem2ViewCalibOpen関数で指定した画像サイズとします．なお，探索

対象画像は，厳密には歪み補正画像を用いますが，精度上問題がなければ原画像でも構いません．また，探索対象画像は，射影変換画像でも構いません（射影変換画像は，歪み補正が施されています）．

*tpObjRect　［入力］探索範囲の矩形領域（左上座標およびサイズ）．

iX：左上X座標（画素）．

iY：左上Y座標（画素）．

iWidth：領域のX方向サイズ（画素）．

iHeight：領域のY方向サイズ（画素）．

*tpEpiLine　［入力］探索対象画像内のエピポーラ線情報（直線の係数のみ使用）．

eyem2ViewCalcEpiLine関数 または eyem2ViewChangeHomEpiLine関数の出力値を指定します．

tLine：エピポーラ線（$ax + by + c = 0$）の係数a, b, c．

taPt[2]：エピポーラ線と画像境界との交点（2点）の座標．

*tpCrspnd　［出力］探索対象画像内の対応点．

dX：X座標（画素）．

dY：Y座標（画素）．

*dpCorr　［出力］対応点サーチ結果の相関値．

iMethod（サーチ方法）に対する，相関値の範囲は以下のとおりです．

iMethod	相関値の範囲
EYEM_2VIEW_SEARCH_ZNCC	-1～1
EYEM_2VIEW_SEARCH_SAD	0～1

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	ワークメモリ不足
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません．

eyem2ViewGetCamHomMat

機 能　ホモグラフィー行列の取得

形 式

#include “eyem2View.h”

int　eyem2ViewGetCamHomMat(void *vpCalib, char cLR, EyemCalibHom *tpHom);

解 説

指定カメラのホモグラフィー行列を取得します．この行列は，0 番目パターン平面（すなわち第1パターン座標系xy面）と画像平面との間の射影変換行列 $\mathbf{H}$ です．すなわち，0 番目パターン上3次元座標を$(X, Y, 0)$，およびレンズ歪みなしの画像座標を(x, y) とするとき，λを実数として，

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

となります．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］左右カメラ画像の選択．
左カメラ画像の場合は ’L’， 右カメラ画像の場合は ’R’ を指定します．

*tpHom　［出力］ホモグラフィー行列．
daH[3][3]： ホモグラフィー行列 $\mathbf{H}$.
daInvH[3][3]： 行列 $\mathbf{H}$ の逆行列 $\mathbf{H}^{-1}$.

戻り値

終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項

特にありません．

eyem2ViewGetCamFndMat

機 能　　基礎行列の取得

形 式

#include “eyem2View.h”

int　eyem2ViewGetCamFndMat(void *vpCalib, Eyem2ViewFnd *tpFnd);

解 説

左右カメラ間の基礎行列を取得します．この行列は，左右の画像点を結び付ける行列 **F** であり，次のエピポーラ方程式を満たしています．すなわち，左カメラの歪みなし画像座標を (x_L, y_L)，および右カメラの歪みなし画像座標を (x_R, y_R) とするとき

$$(x_L \quad y_L \quad 1)\mathbf{F}\begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix} = 0$$

となります．

引 数

*vpCalib　　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

*tpFnd　　［出力］基礎行列．
daF[3][3]： 基礎行列 **F**．
daFT[3][3]： 行列 **F** の転置行列 $\mathbf{F}^{\mathrm{T}}$．

戻り値

終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　　特にありません．

eyem2ViewGetCamIntParam

機 能　カメラ内部パラメータの取得

形 式

```
#include "eyem2View.h"
int    eyem2ViewGetCamIntParam( void *vpCalib, char cLR, EyemCalibInt *tpInt );
```

解 説　指定カメラの内部パラメータを取得します．内部パラメータは，x方向，y方向それぞれの「焦点距離×スケール」(f_xおよびf_y)，「焦点距離×せん断係数」(f_s)，「画像中心」((u_0, v_0))および「半径方向の歪み(ラジアル歪み)係数」が得られます．これらを用いて，内部パラメータ行列 $\mathbf{A}$ は，次の3×3行列で表されます．

$$\mathbf{A} = \begin{pmatrix} f_x & f_s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］左右カメラの選択．
左カメラの場合は 'L'， 右カメラの場合は 'R' を指定します．

*tpInt　［出力］カメラ内部パラメータ．
daA[3][3]：内部パラメータ行列 $\mathbf{A}$．
dFx, dFy：焦点距離×スケール(f_xおよびf_y)(画素)．
dFs：　焦点距離×せん断係数(f_s)(画素)．
dUo, dVo：画像中心座標((u_0, v_0))(画素)．
dK1, dK2：ラジアル歪み係数．
dTheta：画像面傾斜角(CCD面x軸周りの角度)(rad)(シャインプルーフ構成カメラの場合に使用可能．)

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません．

eyem2ViewGetCamExtParam

機 能　カメラ外部パラメータの取得(ワールド座標系基準)

形 式

```
#include "eyem2View.h"
int    eyem2ViewGetCamExtParam( void *vpCalib, char cLR, EyemCalibExt *tpExt );
```

解 説　指定カメラの外部パラメータを取得します. 外部パラメータは, ワールド座標系の3次元座標をカメラ座標へ変換する「回転」および「平行移動」です. なお, ワールド座標系については, **Appendix3** をご参照ください.

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ.
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します.

cLR　［入力］左右カメラの選択.
左カメラの場合は 'L', 右カメラの場合は 'R' を指定します.

*tpExt　［出力］カメラ外部パラメータ.
daR[3][3]: 回転行列.
daT[3]: 平行移動ベクトル.
daRdr[3]: 回転ベクトル(ロドリゲスの表現).

戻り値　終了コードです.

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません.

eyem2ViewGetCamExtParamPtn

機 能　カメラ外部パラメータの取得(パターン座標系基準)

形 式

```
#include "eyem2View.h"
int    eyem2ViewGetCamExtParamPtn( void *vpCalib, char cLR, int iPtnNo,
                                   EyemCalibExt *tpExt );
```

解 説　指定カメラの外部パラメータを取得します. 外部パラメータは, 各パターンをxy平面とする右手座標系の3次元座標をカメラ座標へ変換する「回転」および「平行移動」です.

引 数

*vpCalib　[入力] キャリブレーション・ディスクリプタ.
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します.

cLR　[入力] 左右カメラの選択.
左カメラの場合は 'L', 右カメラの場合は 'R' を指定します.

iPtnNo　[入力] パターンの番号.
0番からの通し番号を入力します. 画像セット番号に対応しています.

*tpExt　[出力] カメラ外部パラメータ.
daR[3][3]: 回転行列.
daT[3]: 平行移動ベクトル.
daRdr[3]: 回転ベクトル(ロドリゲスの表現).

戻り値　終了コードです.

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません.

eyem2ViewGetPtnNum

機 能　パターン・ドットの画像座標データ個数の取得

形 式

```
#include "eyem2View.h"
int    eyem2ViewGetPtnNum( void *vpCalib, char cLR, int iDataNo, int *ipPtnNum );
```

解 説　指定カメラのキャリブレーションで使用した，パターン・ドットの画像座標データの個数を取得します．

引 数

*vpCalib	［入力］キャリブレーション・ディスクリプタ． eyem2ViewCalibOpen関数の出力値を指定します．
cLR	［入力］左右カメラの選択． 左カメラの場合は 'L'， 右カメラの場合は 'R' を指定します．
iDataNo	［入力］座標データの番号． 0番からの通し番号を入力します．画像セット番号に対応しています．
*ipPtnNum	［出力］パターン・ドットの画像座標データの個数．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可（データなし）
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　本関数は，eyem2ViewCalibOpen関数でのオープン時に限り使用可能となります．

eyem2ViewGetPattern

機 能　パターン・ドットの画像座標データの取得

形 式

```
#include "eyem2View.h"
int　eyem2ViewGetPattern( void *vpCalib, char cLR, int iDataNo, EyemOcsDXY taPtn[] );
```

解 説　指定カメラのキャリブレーションで使用した，パターン・ドット点の画像座標データを取得します．予め，eyem2ViewGetPtnNum 関数で取得した個数分の配列 taPtn[] を確保してください．

引 数

*vpCalib	［入力］キャリブレーション・ディスクリプタ． eyem2ViewCalibOpen関数の出力値を指定します．
cLR	［入力］左右カメラの選択． 左カメラの場合は 'L'， 右カメラの場合は 'R' を指定します．
iDataNo	［入力］座標データの番号． 0番からの通し番号を入力します．画像セット番号に対応しています．
taPtn[]	［出力］パターン・ドット点の画像座標データ． dX： X座標（画素）． dY： Y座標（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可（データなし）
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　本関数は，eyem2ViewCalibOpen関数でのオープン時に限り使用可能となります．

eyem2ViewMakeHomImage

機 能　画像の射影変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewMakeHomImage( void *vpCalib, char cLR, unsigned char *ucpImgSrc,
                                    Eyem2ViewHom *tpParam, unsigned char *ucpImgDst );
```

解 説　対応点探索用の画像として，指定カメラの原画像を射影変換し，もう一方のカメラから見たような画像にします．なお，画像射影変換については，**Appendix5** をご参照ください．

引 数

*vpCalib	［入力］キャリブレーション・ディスクリプタ． eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．
cLR	［入力］左右カメラ画像の選択． 左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．
*ucpImgSrc	［入力］原画像メモリの先頭アドレス． eyem2ViewCalibOpen関数で指定した画像サイズとします．
*tpParam	［入力］画像射影変換パラメータ． tCenter：射影変換画像の中央に対応する，原画像の座標点． dWorldZ：計測点3次元座標の*z*座標概略値．
*ucpImgDst	［出力］射影変換画像を格納する画像メモリの先頭アドレス． eyem2ViewCalibOpen関数で指定した画像サイズとします．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません．

eyem2ViewChangeHom2Dto3D

機 能　左右カメラ画像（一方が射影変換画像）の対応点から3次元座標点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewChangeHom2Dto3D( void *vpCalib, int iImgL, EyemOcsDXY *tpImgPtL,
                                 int iImgR, EyemOcsDXY *tpImgPtR, Eyem2ViewHom *tpParam,
                                 EyemOcsDXYZ *tpPt3D );
```

解 説　左右カメラ画像（一方が射影変換画像）における対応点から，その3次元座標を求めます．なお，射影変換画像は，eyem2ViewMakeHomImage 関数で作成された画像とします．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

iImgL　［入力］左カメラの画像種別．
以下のいずれかを指定します．

EYEM_2VIEW_IMG_ACTUAL	原画像（レンズ歪みのある画像）
EYEM_2VIEW_IMG_IDEAL	理想画像（レンズ歪み補正画像）
EYEM_2VIEW_IMG_HOM	射影変換画像

*tpImgPtL　［入力］左カメラ画像における座標．
dX： X座標（画素）．
dY： Y座標（画素）．

iImgR　［入力］右カメラの画像種別．
以下のいずれかを指定します．

EYEM_2VIEW_IMG_ACTUAL	原画像（レンズ歪みのある画像）
EYEM_2VIEW_IMG_IDEAL	理想画像（レンズ歪み補正画像）
EYEM_2VIEW_IMG_HOM	射影変換画像

*tpImgPtR　［入力］右カメラ画像における座標．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpParam　［入力］画像射影変換パラメータ．
必ず，eyem2ViewMakeHomImage関数で用いた値を指定します．なお，画像射影変換については，**Appendix5** をご参照ください．
tCenter： 射影変換画像の中央に対応する，原画像の座標点．
dWorldZ： 計測点3次元座標の*z*座標概略値．

*tpPt3D　［出力］3次元座標．
dX： X座標（mm）．

dY： Y座標(mm).

dZ： Z座標(mm).

戻り値　終了コードです.

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項　特にありません.

eyem2ViewInvHomImgPoint

機 能　射影変換画像点から原画像点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewInvHomImgPoint( void *vpCalib, char cLR, EyemOcsDXY *tpPtSrc,
                                Eyem2ViewHom *tpParam, EyemOcsDXY *tpPtDst );
```

解 説　指定カメラの射影変換画像上の座標点を，原画像上の座標点に変換します．なお，射影変換画像は，eyem2ViewMakeHomImage 関数で作成された画像とします．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］左右カメラ画像の選択．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*tpPtSrc　［入力］射影変換画像上の座標点．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpParam　［入力］画像射影変換パラメータ．
必ず，eyem2ViewMakeHomImage関数で用いた値を指定します．なお，画像射影変換については，**Appendix5** をご参照ください．
tCenter： 射影変換画像の中央に対応する，原画像の座標点．
dWorldZ： 計測点3次元座標のz座標概略値．

*tpPtDst　［出力］原画像上の座標点．
tpPtSrc と同じでも構いません．
dX： X座標（画素）．
dY： Y座標（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません．

eyem2ViewCalcHomImgPoint

機 能　原画像点から射影変換画像点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewCalcHomImgPoint( void *vpCalib, char cLR, EyemOcsDXY *tpPtSrc,
                                 Eyem2ViewHom *tpParam, EyemOcsDXY *tpPtDst );
```

解 説　指定カメラの原画像上の点を，その射影変換画像上の座標点に変換します．なお，射影変換画像は，eyem2ViewMakeHomImage 関数で作成された画像とします．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］左右カメラ画像の選択．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*tpPtSrc　［入力］原画像上の座標点．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpParam　［入力］画像射影変換パラメータ．
必ず，eyem2ViewMakeHomImage関数で用いた値を指定します．なお，画像射影変換については，**Appendix5** をご参照ください．
tCenter： 射影変換画像の中央に対応する，原画像の座標点．
dWorldZ： 計測点3次元座標の*z*座標概略値．

*tpPtDst　［出力］射影変換画像上の座標点．
tpPtSrc と同じでも構いません．
dX： X座標（画素）．
dY： Y座標（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_CALC	計算不可
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項　特にありません．

eyem2ViewUnDistortImage

機 能　画像の歪み補正

形 式

```
#include “eyem2View.h”
int    eyem2ViewUnDistortImage( void *vpCalib, char cLR, unsigned char *ucpImgSrc,
                                unsigned char *ucpImgDst );
```

解 説　指定カメラの原画像に対して，歪み補正した画像を作成します．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］左右カメラ画像の選択．
左カメラ画像の場合は ’L’， 右カメラ画像の場合は ’R’ を指定します．

*ucpImgSrc　［入力］原画像メモリの先頭アドレス．
eyem2ViewCalibOpen関数で指定した画像サイズとします．

*ucpImgDst　［出力］歪み補正画像を格納する画像メモリの先頭アドレス．
eyem2ViewCalibOpen関数で指定した画像サイズとします．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　特にありません．

eyem2ViewDistortPoint

機 能　　歪み補正画像点から原画像点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewDistortPoint( void *vpCalib, char cLR, EyemOcsDXY *tpPtSrc,
                              EyemOcsDXY *tpPtDst );
```

解 説　　指定カメラの歪み補正画像上の座標点を，原画像上の座標点に変換します．すなわち，座標点に歪みを加えます．なお，歪み補正画像は，eyem2ViewUndistortImage 関数で作成された画像とします．

引 数

*vpCalib　　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　　［入力］左右カメラ画像の選択．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*tpPtSrc　　［入力］歪み補正画像上の座標点．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpPtDst　　［出力］原画像上の座標点．
tpPtSrc と同じでも構いません．
dX： X座標（画素）．
dY： Y座標（画素）．

戻り値　　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可

留意事項　　特にありません．

eyem2ViewUnDistortPoint

機 能　原画像点から歪み補正画像点への変換

形 式

```
#include "eyem2View.h"
int    eyem2ViewUnDistortPoint( void *vpCalib, char cLR, EyemOcsDXY *tpPtSrc,
                                EyemOcsDXY *tpPtDst );
```

解 説　指定カメラの原画像上の座標点を，歪み補正画像上の座標点に変換します．すなわち，座標点の歪みを除去します．なお，歪み補正画像は，eyem2ViewUndistortImage 関数で作成された画像とします．

引 数

*vpCalib　［入力］キャリブレーション・ディスクリプタ．
eyem2ViewCalibOpen関数 または eyem2ViewLoadCalibDataF関数の出力値を指定します．

cLR　［入力］左右カメラ画像の選択．
左カメラ画像の場合は 'L'， 右カメラ画像の場合は 'R' を指定します．

*tpPtSrc　［入力］原画像上の座標点．
dX： X座標（画素）．
dY： Y座標（画素）．

*tpPtDst　［出力］歪み補正画像上の座標点．
tpPtSrc と同じでも構いません．
dX： X座標（画素）．
dY： Y座標（画素）．

戻り値　終了コードです．

FUNC_OK	正常終了
FUNC_ILLEGAL_ARGUMENT	引数が不適当
FUNC_CANNOT_USE	ライブラリ使用不可
FUNC_FAILED_UNDISTORT	歪み補正失敗

留意事項　特にありません．

Appendix1. 本ライブラリで使用している定数および構造体

本ライブラリで使われている定数および構造体の具体的内容は以下のとおりです.

```
// エラーコード
#define FUNC_OK                          0        // 正常終了
#define FUNC_NOT_ENOUGH_MEM              (-1)     // ワークメモリ不足
#define FUNC_ILLEGAL_ARGUMENT            (-2)     // 引数が不適当
#define FUNC_CANNOT_CALC                 (-100)   // 計算不可
#define FUNC_FAILED_HOMOGRAPHY           (-150)   // ホモグラフィー行列計算失敗
#define FUNC_FAILED_CAM_PRM              (-151)   // カメラパラメータ計算失敗
#define FUNC_FAILED_BUNDLE_ADJ           (-152)   // バンドル調整失敗
#define FUNC_FAILED_UNDISTORT            (-153)   // 歪み補正失敗
#define FUNC_FEW_PTN_SMPL_NUM            (-200)   // パターンの参照点数が足りない
#define FUNC_CANNOT_READ_MARK            (-201)   // パターンマーカー点(大ドット点)の認識不可
#define FUNC_CANNOT_READ_PTN             (-202)   // パターン参照点の実座標の認識不可
#define FUNC_FAILED_FUNDAMENTAL          (-210)   // 基礎行列計算失敗
#define FUNC_CANNOT_FIND_PLATE           (-220)   // キャリブレーションプレートが見つからない


// カメラ種別
enum {
        EYEM_LSM_CAM_NORMAL          = 0,     // 通常のカメラ
        EYEM_LSM_CAM_SCHEIMPFLUG     = 1      // シャインプルーフ構成のカメラ
        EYEM_2VIEW_CAM_TELECENTRIC   = 2      // テレセントリックレンズ使用のカメラ
};


// ワールド座標系種別
enum {
        EYEM_2VIEW_WLD_PTN,                   // 第1パターン座標系
        EYEM_2VIEW_WLD_CAM_LEFT,              // 左カメラ座標系
        EYEM_2VIEW_WLD_CAM_RIGHT,             // 右カメラ座標系
        EYEM_2VIEW_WLD_CAM_MIDDLE             // 中間点座標系(左右カメラの中間点を原点とする座標系)
};


// 画像種別
enum {
        EYEM_2VIEW_IMG_ACTUAL,                // 原画像(レンズ歪みのある画像)
        EYEM_2VIEW_IMG_IDEAL,                 // 理想画像(レンズ歪み補正された画像)
        EYEM_2VIEW_IMG_HOM                    // 射影変換画像(eyem2ViewMakeHomImage 関数の出力画像)
};


// サーチ類似度計算方法
enum {
        EYEM_2VIEW_SEARCH_ZNCC,               // ゼロ平均正規化相互相関(ZNCC)法
        EYEM_2VIEW_SEARCH_SAD                 // 差分絶対総和(SAD)法
};
```

```
// キャリブレーション治具情報
typedef struct {
        double      dPatternSizeX;          // パターン横方向(x方向)サイズ(mm)(パターン範囲の幅)
        double      dPatternSizeY;          // パターン縦方向(y方向)サイズ(mm)(パターン範囲の高さ)
        double      dDotSizeSmall;          // 小ドットサイズ(直径)(mm)
        double      dDotSizeLarge;          // 大ドットサイズ(直径)(mm)
        double      dDotPitchSmallX;        // 小ドット横方向(x方向)ピッチ(mm)
        double      dDotPitchSmallY;        // 小ドット縦方向(y方向)ピッチ(mm)
        double      dDotPitchLargeX;        // 大ドット横方向(x方向)ピッチ(mm)
        double      dDotPitchLargeY;        // 大ドット縦方向(y方向)ピッチ(mm)
        int         iLargeDotNum;           // 大ドット個数

        int         iDotColor;              // (2値化)パターンドットの色(黒:EYEM_BIN_BLACK, 白:EYEM_BIN_WHITE)
        int         iDotAreaThrs;           // (2値化)パターンドット点の面積下限値(画素)(ノイズ除去用)
        double      dDotEdgeStrThrs;        // パターンドット点のエッジ強度下限値(0 以上)(ぼけドット除去用, 標準値:0)
        int         iMinPtnPtNum;           // パターン参照点最小数( max( 4, iLargeDotNum ) 以上)
}       Eyem2ViewJig;


// カメラ情報
typedef struct {
        int         iCamL;                  // 左カメラ種別(上記 enum 定数)
        int         iWidthL;                // 左カメラの画像メモリx方向サイズ
        int         iHeightL;               // 左カメラの画像メモリy方向サイズ
        int         iCamR;                  // 右カメラ種別(上記 enum 定数)
        int         iWidthR;                // 右カメラの画像メモリx方向サイズ
        int         iHeightR;               // 右カメラの画像メモリy方向サイズ
}       Eyem2ViewCam;


// ステレオ画像セット
typedef struct {
        unsigned char       *ucpImgCamL;    // 左カメラの画像メモリ先頭アドレス
        unsigned char       *ucpImgCamR;    // 右カメラの画像メモリ先頭アドレス
}       Eyem2ViewImg;


// バンドル調整の結果
typedef struct {
        int         iNumL;                  // 左カメラのバンドル調整反復回数
        double      dRmsL;                  // 左カメラのバンドル調整残差 RMS(平均二乗誤差の平方根)(単位:画素)
        int         iNumR;                  // 右カメラのバンドル調整反復回数
        double      dRmsR;                  // 右カメラのバンドル調整残差 RMS(平均二乗誤差の平方根)(単位:画素)
}       Eyem2ViewBndl;


// 基礎行列
typedef struct {
        double      daF[3][3];              // 基礎行列 F( (左画像点)＊F＊(右画像点) = 0 )
        double      daFT[3][3];             // 基礎行列 F の転置行列
}       Eyem2ViewFnd;


// エピポーラ線情報
typedef struct {
        EyemOcsDABC         tLine;          // エピポーラ線の係数(ax+by+c=0)
        EyemOcsDXY          taPt[2];        // エピポーラ線と画像境界との交点(2点)
}       Eyem2ViewEpi;
```

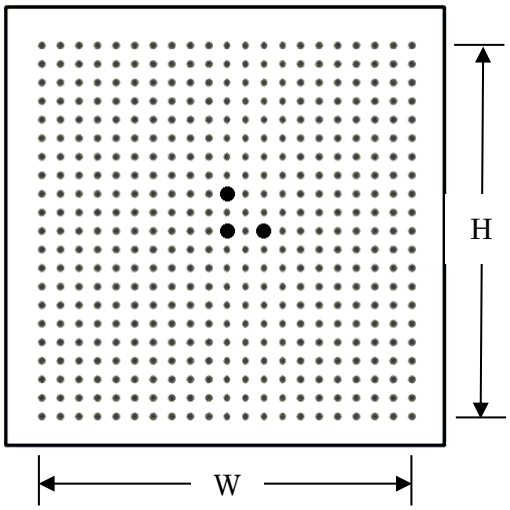
```
// 画像射影変換パラメータ
typedef struct {
        EyemOcsDXY      tCenter;    // 射影変換画像の中点座標(画像幅/2, 画像高さ/2)に対応する, 原画像の座標点
                                    // (計測点の近傍が望ましい)
        double          dWorldZ;    // ( = h )第1パターン座標系における, 計測点のz座標概略値
                                    // (他方のカメラに平面 z=h を投影したときの画像寸法に合わせるように射影変換する)
}       Eyem2ViewHom;


// パターンデータ
typedef struct {
        int             iPtnRefN;           // パターン参照点の個数(4点以上)
        EyemOcsDXY      *tpObjPt;           // パターン参照点の実座標(iPtnRefN 個の配列, 単位:mm)
        EyemOcsDXY      *tpImgPt;           // パターン参照点の画像座標(iPtnRefN 個の配列, 単位:画素)
}       EyemCalibPtn;


// ホモグラフィー行列
typedef struct {
        double  daH[3][3];                  // ホモグラフィー行列 H
        double  daInvH[3][3];               // 行列 H の逆行列
}       EyemCalibHom;


// カメラの内部パラメータ(intrinsic parameter)
typedef struct {
        double  daA[3][3];                  // 内部パラメータ行列
        double  dFx, dFy;                   // 焦点距離×スケール(単位:画素)
        double  dFs;                        // 焦点距離×せん断係数(単位:画素)
        double  dUo, dVo;                   // 画像中心座標(単位:画素)
        double  dTheta;                     // 画像面傾斜角(単位:rad)
        double  dK1, dK2;                   // ラジアル歪み係数
        double  daRsv[5];                   // 予備領域
}       EyemCalibInt;


// カメラの外部パラメータ(extrinsic parameter)
typedef struct {
        double  daR[3][3];                  // 回転行列
        double  daT[3];                     // 並進ベクトル
        double  daRdr[3];                   // 回転ベクトル(ロドリゲスの表現)
}       EyemCalibExt;


// 矩形定義
typedef struct {
        int     iXs;                        // 始点(左上)x座標
        int     iYs;                        // 始点(左上)y座標
        int     iWidth;                     // x方向サイズ(幅)
        int     iHeight;                    // y方向サイズ(高さ)
}       EyemRect;


// 直線定義(平面上:ax+by+c=0)
typedef struct{
        double  dA;                         // a
        double  dB;                         // b
        double  dC;                         // c
}       EyemOcsDABC;
```

```
// 座標系定義(double 型)
typedef struct{
        double      dX;                     // X座標
        double      dY;                     // Y座標
}       EyemOcsDXY;
```

Appendix2. キャリブレーション治具情報構造体 Eyem2ViewJig の設定

本ライブラリでは，下図に示すような大小の円形ドットパターンで構成されたキャリブレーション治具を想定しています．すなわち，格子点に小ドットを配置し，中央部分に座標軸を認識するための大ドットを配置します．大ドットについては，まず座標原点としてプレート中央に1つ配置し，さらに x 軸上および y 軸上にそれぞれ1つずつ，合わせて3つを配置します．

このキャリブレーション治具の情報を設定する Eyem2ViewJig 構造体の具体的内容は以下のとおりです．



dPatternSizeX：パターン横方向（x方向）サイズ（mm）．
左図の W サイズです．

dPatternSizeY：パターン縦方向（y方向）サイズ（mm）．
左図の H サイズです．

dDotSizeSmall：小ドットサイズ（直径）（mm）．

dDotSizeLarge：大ドットサイズ（直径）（mm）．

dDotPitchSmallX：小ドット横方向（x方向）間隔（mm）．

dDotPitchSmallY：小ドット縦方向（y方向）間隔（mm）．

dDotPitchLargeX：大ドット横方向（x方向）間隔（mm）．

dDotPitchLargeY：大ドット縦方向（y方向）間隔（mm）．

iLargeDotNum：大ドット個数．
現状は 3 を指定します．

iDotColor：ドットの色．
2値化におけるドットの色です．EYEM_BIN_BLACK（黒） または EYEM_BIN_WHITE（白） のいずれかを指定します．

iDotAreaThrs：ドットの面積下限値（画素）．
2値化におけるノイズ除去のための値です．指定値以上の面積をもつ2値ブロッブが処理対象となります．

dDotEdgeStrThrs：ドットのエッジ強度下限値．
ピントぼけしたドットを除去するための値です．0 以上の値を指定します（標準値:0）．

iMinPtnPtNum：ドット最小数．
認識すべきドット個数の最小値を max(4, iLargeDotNum) 以上の値で指定します．

Appendix3. ワールド座標系

本ライブラリで使用できるワールド座標系は以下の4種類で，すべて右手座標系です．

（1） 第1パターン座標系

キャリブレーションに用いたパターン群における1枚目（インデックス 0 番）のパターン面を xy 面とする座標系です．

（2） 左カメラ座標系

左カメラのレンズ中心を原点とし，光軸を z 軸とした座標系です．

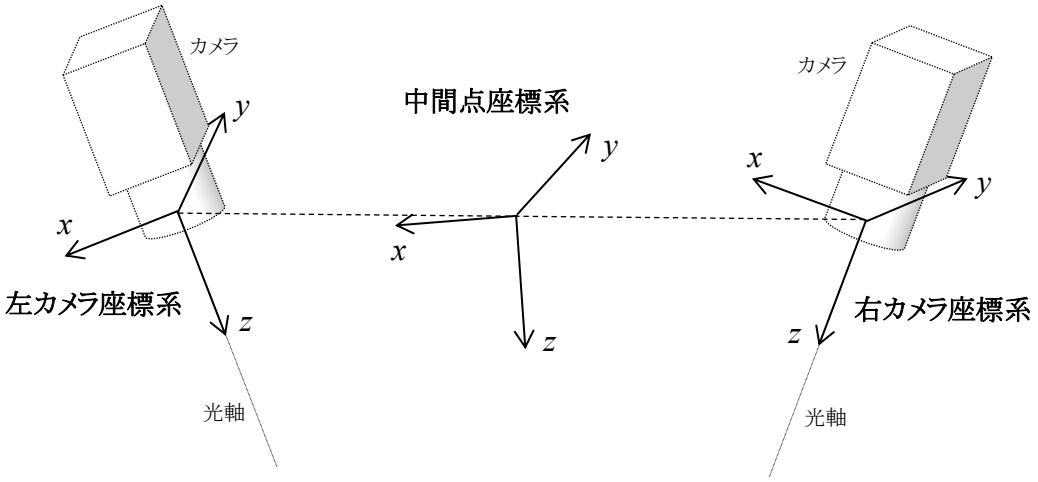
（3） 右カメラ座標系

右カメラのレンズ中心を原点とし，光軸を z 軸とした座標系です．

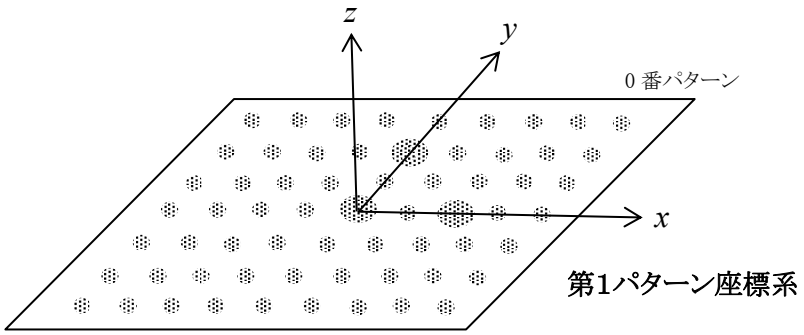
（4） 中間点座標系

左右レンズ中心の中間点を原点とした座標系です．座標系の各軸の基底ベクトルは，左右カメラ座標系のそれぞれの軸の基底ベクトルの平均としています．

※ この他に，ワールド座標系ではないが，キャリブレーションに用いた各パターンに付随する，第1パター座標系と同じ設定の座標系を，パターン座標系とします．



」



Appendix4. キャリブレーションの方法

キャリブレーションの具体的な手順は以下のとおりです.

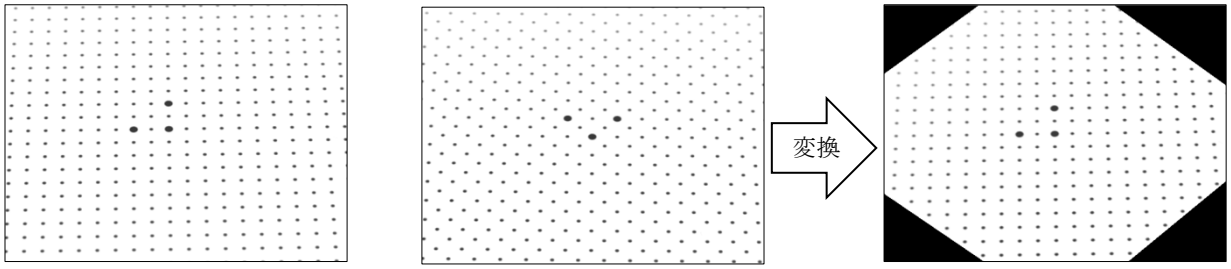
① 第1パターン座標系の xy 面を設定したい場所にパターンを設置します.
② パターンを固定したまま, 左右のカメラそれぞれで画像を読み込みます. これを, 1セットとします.
③ パターンの位置および向きを変えながら②の手順を繰り返し, 画像セットを3セット以上取得します.
④ 本キャリブレーションライブラリを実行します.

留意事項

画像内に映るパターンが画像内をまんべんなく, さらに, どの2枚も平行とならないようにパターンの位置と向きを調整してください. すなわち, 全体として, パターン・ドットが画像内にまんべんなく配置されるようにしてください. これは, 歪み補正係数を求めるために必要な措置です.

Appendix5. 画像射影変換について

画像射影変換では，第1パターン座標系内の平面 $z = h$（h は，Eyem2ViewHom 構造体のパラメータ指定値）の見え方を同じにします．具体的には，一方のカメラ（A とします）から見た平面 $z = h$ の画像と同じになるように，他方のカメラ（B とします）から見た平面 $z = h$ の画像を変換します．



カメラ A 原画像　　カメラ B 原画像　　カメラ B 射影変換画像

注意していただきたいのは，あくまで平面 $z = h$ に対して同じ見え方にするということです．例えばカメラ B の画像に平面 $z = h$ 上にない対象 α が映っている場合は，その α が平面 $z = h$ 上にある，として変換をしてしまいます．よって，平面 $z = h$ 上にない対象物については，カメラ A 画像と変換後のカメラ B 画像において，画像内の対象物の相対的な位置関係が崩れてしまうので，それらを計測対象にすると正確な3次元位置を求めることができません．

画像射影変換構造体 Eyem2ViewHom のパラメータ設定は，以下のとおりです．

- EyemOcsDXY　tCenter

  射影変換画像の中点座標（画像幅/2，画像高さ/2）に対応する，原画像の座標点です．

  dX：X座標（画素）．

  dY：Y座標（画素）．

- double　dWorldZ

  第1パターン座標系における，計測点のz座標概略値です．上記の h です．

改訂履歴

Version No.	日付	内　　容
1.0	2015.08.22	・新規発行
1.1	2015.08.24	・誤記の修正（eyem2ViewReadPtnImg）
1.2	2016.06.10	・エラーコード（FUNC_CANNOT_USE）の新規追加
2.0	2018.01.31	［ワールド座標系追加に伴う改訂］ ・eyem2ViewCalibOpen 関数にワールド座標系選択の引数追加 ・カメラ外部パラメータ取得関数を基準座標系により分割 （eyem2ViewGetCamExtParam，eyem2ViewGetCamExtParamPtn） ・Appendix1：ワールド座標系種別マクロの追加 ・Appendix3：ワールド座標系の追加 ・その他，文言の微修正
2.1	2018.07.25	・ライブラリ使用手順のキャリブレーションフェーズの追加修正 ・eyem2ViewCalibOpen 関数の治具引数に NULL 指定を追加 ・eyem2ViewCalibCamPrmWithUserJig 関数の新規追加 ・Appendix1：パターンデータ構造体の追加 ・その他，文言の微修正
2.2	2018.08.08	・説明文の追加（eyem2ViewCalibCamPrmWithUserJig） ・説明文の修正（eyem2ViewCorrespondence）
2.3	2018.08.21	・ライブラリ使用手順の計測フェーズの修正 ・関数の新規追加 （eyem2ViewChange2Dto3D2，eyem2ViewCalcEpiLine2， eyem2ViewCorrespondence2，eyem2ViewChangeHom2Dto3D） ・Appendix1：画像種別マクロの追加

以　上