static Mat prepareCameraMatrix(Mat& cameraMatrix0, int rtype)

{

Mat cameraMatrix = Mat::eye(3, 3, rtype);

if( cameraMatrix0.size() == cameraMatrix.size() )

cameraMatrix0.convertTo(cameraMatrix, rtype);

return cameraMatrix;

}

将数据类型转换到指定的类型

static void collectCalibrationData( InputArrayOfArrays objectPoints,

InputArrayOfArrays imagePoints1,

InputArrayOfArrays imagePoints2,

Mat& objPtMat, Mat& imgPtMat1, Mat\* imgPtMat2,

Mat& npoints )

{

int nimages = (int)objectPoints.total();

int i, j = 0, ni = 0, total = 0;

CV\_Assert(nimages > 0 && nimages == (int)imagePoints1.total() &&

(!imgPtMat2 || nimages == (int)imagePoints2.total()));

for( i = 0; i < nimages; i++ )

{

ni = objectPoints.getMat(i).checkVector(3, CV\_32F);

if( ni <= 0 )

CV\_Error(CV\_StsUnsupportedFormat, "objectPoints should contain vector of vectors of points of type Point3f");

int ni1 = imagePoints1.getMat(i).checkVector(2, CV\_32F);

if( ni1 <= 0 )

CV\_Error(CV\_StsUnsupportedFormat, "imagePoints1 should contain vector of vectors of points of type Point2f");

CV\_Assert( ni == ni1 );

total += ni;

}

npoints.create(1, (int)nimages, CV\_32S);

objPtMat.create(1, (int)total, CV\_32FC3);

imgPtMat1.create(1, (int)total, CV\_32FC2);

Point2f\* imgPtData2 = 0;

if( imgPtMat2 )

{

imgPtMat2->create(1, (int)total, CV\_32FC2);

imgPtData2 = imgPtMat2->ptr<Point2f>();

}

Point3f\* objPtData = objPtMat.ptr<Point3f>();

Point2f\* imgPtData1 = imgPtMat1.ptr<Point2f>();

for( i = 0; i < nimages; i++, j += ni )

{

Mat objpt = objectPoints.getMat(i);

Mat imgpt1 = imagePoints1.getMat(i);

ni = objpt.checkVector(3, CV\_32F);

npoints.at<int>(i) = ni;

for (int n = 0; n < ni; ++n)

{

objPtData[j + n] = objpt.ptr<Point3f>()[n];

imgPtData1[j + n] = imgpt1.ptr<Point2f>()[n];

}

if( imgPtData2 )

{

Mat imgpt2 = imagePoints2.getMat(i);

int ni2 = imgpt2.checkVector(2, CV\_32F);

CV\_Assert( ni == ni2 );

for (int n = 0; n < ni2; ++n)

{

imgPtData2[j + n] = imgpt2.ptr<Point2f>()[n];

}

}

}

}

nimages 图像张数

total 点的总数

npoints 1行nimages列 表示每张图像多少个点

objPtMat 1 行 total 列 将所有三维点并排在一起

imgPtMat1 1 行 total 列 将所有相机1观察到的二维点并排在一起

imgPtMat2 1 行 total 列 将所有相机2观察到的二维点并排在一起

c\_objPt c\_imgPt c\_imgPt2 c\_npoints

c\_cameraMatrix1 c\_cameraMatrix2

c\_distCoeffs1 c\_distCoeffs2

c\_matR c\_matT c\_matE c\_matF c\_matErr

数据初始化步骤:

1. prepareCameraMatrix将所有数据转换到指定类型CV\_64F
2. collectCalibrationData
3. Mat转成CvMat

cvStereoCalibrateImpl

objectPoints 3行1列

imagePoints[0], imagePoints[1] 2行1列

K[0]= c\_cameraMatrix1 K[1]= c\_cameraMatrix2

畸变参数为5个

maxPoints 最大点数,每张图像点数可能不一样。

const int NINTRINSIC = 18;

Mat err( maxPoints\*2, 1, CV\_64F );

Mat Je( maxPoints\*2, 6, CV\_64F );

Mat J\_LR( maxPoints\*2, 6, CV\_64F );

Mat Ji( maxPoints\*2, NINTRINSIC, CV\_64F, Scalar(0) );

两个相机之间的R,T关系初始值用轴角表示法，构造一个矩阵为6x nimages



对每一列进行快速排序，取中位数。

每个相机取18个内参fx fy cx cy 14个畸变

(nimages+1)\*6参数说明：

头6个参数是两个相机之间的R、T关系(相机1到相机2)

nimages\*6是世界坐标系到相机1的nimages 张图像的R、T关系

参数个数nparams=(nimages+1)\*6+18\*2





static void subMatrix(const cv::Mat& src, cv::Mat& dst, const std::vector<uchar>& cols,

const std::vector<uchar>& rows) {

int nonzeros\_cols = cv::countNonZero(cols);

cv::Mat tmp(src.rows, nonzeros\_cols, CV\_64FC1);

for (int i = 0, j = 0; i < (int)cols.size(); i++)

{

if (cols[i])

{

src.col(i).copyTo(tmp.col(j++));

}

}

int nonzeros\_rows = cv::countNonZero(rows);

dst.create(nonzeros\_rows, nonzeros\_cols, CV\_64FC1);

for (int i = 0, j = 0; i < (int)rows.size(); i++)

{

if (rows[i])

{

tmp.row(i).copyTo(dst.row(j++));

}

}

}

Opencv根据mask 从JtJ提取出JtJN，从JtErr提取出JtJV

JtErr本来就只有一列所以根据mask抽取行即可。mask是vector形式。

参数更新时也是根据mask更新。