EOL\_cb

1. 算法介绍

本算法参考OCamcalib棋盘角点检测算法，该算法的主要原理是通过自适应阈值分割获得棋盘格的四边形框，然后生成四边形群，对四边形群进行判断。本算法在此基础上做了改进和延伸，主要修改点见以下说明：

1. 检测4\*4、4\*3、4\*2模式基本与原来算法一致，部分模块和参数有修改，并且每次只能检测一种模式
2. 检测1\*1模式只用其中的部分模块，差异点见图3
3. 加入了处理反光的步骤（腐蚀膨胀）
4. 在循环体中加入腐蚀操作，解决了左右相机1\*1模式经常无法检测出的问题
5. 相机检测参数可配置化
6. 主要接口

int32\_t EOL\_Process(IplImage\* sProcessImg\_in[4],Smc\_Cal\_T\* pSMC,eol\_mode mode,eol\_data\_input \*data\_input)

int cvFindChessboardCorners3( const void\* arr);

int cvFindChessboardCorners2( const void\* arr );

1. 数据类型

typedef struct

{

CvSize board\_size; \\板子尺寸模式4\*4,4\*3,4\*2

int min\_number\_of\_corners; \\最小corner数目，一般是板子模式长宽积

int rect\_mask\_num; \\掩膜个数

int reference\_pointNum; \\参考点数目

int camid; \\相机id

float width\_ratio; \\1\*1模式棋盘滤波位置参数

float height\_ratio; \\1\*1模式棋盘滤波位置参数

float RectMinAreaRatio; \\1\*1模式棋盘滤波面积参数

int reflection; \\是否使能处理反光操作

int gloabal\_min\_dilate; [\\最小膨胀指数，0](file:///\\最小膨胀指数，0)是保持原图，负数是腐蚀操作

}chessBoard\_corner\_config;

typedef struct

{

IplImage \*img\_in; \\输入图像指针

int elem\_size; \\根据板子模式分配的点空间大小

CvSeq\* image\_points\_seq; \\点序列

CvPoint2D32f\* image\_points\_buf; \\点暂存区

CvMemStorage\* storage\_max; [\\ opencv](file:///\\opencv)内存区

Rect rect\_mask\_roi[5]; \\掩膜位置

Point2f reference\_point[5]; \\参考点

chessBoard\_pattern pattern\_corner; \\输出的4\*4,4\*3,4\*2 pattern

chessBoard\_pattern pattern\_Single\_corner[2]; [\\输出的两个1\*1](file:///\\输出的两个1*1) pattern

chessBoard\_corner\_config corner\_config; \\输入的检测配置参数

int corner\_count; \\内部变量

void init(); \\初始化

void deinit(); \\去初始化

int cvFindChessboardCorners3( const void\* arr); [\\4\*4,4\*3](file:///\\4*4,4*3),4\*2棋盘检测函数

int cvFindChessboardCorners2( const void\* arr ); [\\1\*1](file:///\\1*1)棋盘检测

int icvGenerateQuads( CvCBQuad \*\*quads, CvCBCorner \*\*corners,

CvMemStorage \*storage, CvMat \*image, int flags, int dilation,bool firstRun ); \\棋盘中产生多边形

void mrFindQuadNeighbors2( CvCBQuad \*quads, int quad\_count, int dilation);

\\寻找相邻的多边形

int mrAugmentBestRun( CvCBQuad \*new\_quads, int new\_quad\_count, int new\_dilation,

CvCBQuad \*\*old\_quads, int old\_quad\_count, int old\_dilation ); \\查找遗漏的多边形群

int icvFindConnectedQuads( CvCBQuad \*quads, int quad\_count, CvCBQuad \*\*quad\_group,

int group\_idx,

CvMemStorage\* storage, int dilation );

\\查找多边形群

void mrLabelQuadGroup( CvCBQuad \*\*quad\_group, int count, CvSize pattern\_size,

bool firstRun );

\\标记多边形群

void mrCopyQuadGroup( CvCBQuad \*\*temp\_quad\_group, CvCBQuad \*\*out\_quad\_group,

int count );

int icvCleanFoundConnectedQuads( int quad\_count, CvCBQuad \*\*quads,

CvSize pattern\_size );

\\滤除不符合pattern尺寸的多边形群

int mrWriteCorners( CvCBQuad \*\*output\_quads, int count, CvSize pattern\_size,

int min\_number\_of\_corners );

\\输出最后的corner群位置

void PointLine\_dist(Point2f point\_in,Point2f line\_start,Point2f line\_end,float &dist );

}chessBoard\_corner\_detector;

typedef struct

{

Point2f corner\_point[CB\_MAX\_ROW][CB\_MAX\_COL];

Point2f ground\_corner\_point[CB\_MAX\_ROW][CB\_MAX\_COL];

int flag[CB\_MAX\_ROW][CB\_MAX\_COL];

int rows;

int cols;

}chessBoard\_pattern; //通用pattern类型

typedef struct

{

chessBoard\_pattern pattern11\_left;

chessBoard\_pattern pattern11\_right;

chessBoard\_pattern pattern42;

chessBoard\_pattern pattern43;

}CbPattern\_group1; //左右相机pattern

typedef struct

{

chessBoard\_pattern pattern11\_left;

chessBoard\_pattern pattern11\_right;

chessBoard\_pattern pattern44;

}CbPattern\_group2; //前后相机pattern

1. Corner检测接口实现
2. 初始化

avm\_landmark以及棋盘检测参数初始化

1. 棋盘检测流程

其中分别对四个相机图像中棋盘进行检测（见图1），其中前后相机棋盘模式为4\*4和两个1\*1，左右相机棋盘模式为4\*3、4\*2和两个1\*1，4\*4、4\*2和4\*3模式都是采用cvFindChessboardCorners3函数进行检测（见图2），1\*1模式采用cvFindChessboardCorners2函数进行检测（见图3）。



图1 总流程



图2 棋盘pattern检测



图3 棋盘pattern检测

**前后相机检测**：先检测4\*4，然后得到一个掩膜和参考点用于检测1\*1，检测调用函数必须都要先初始化，完成后必须deinit；

**左右相机检测**：先检测4\*3，然后得到一个掩膜用于检测4\*2，最后得到两个掩膜和两个参考点用于检测1\*1，检测调用函数必须都要先初始化，完成后必须deinit；

1. 棋盘检测函数说明

init()和deinit()是函数初始和去初始化操作；

cvFindChessboardCorners3

1. 自适应阈值分割（cvAdaptiveThreshold）
2. 对二值化后的图进行掩膜操作
3. 处理光照问题（reflection===1）
4. 循环体（形态学操作、找多边形、找多边形群、滤除不符合条件多边形群、加入遗漏的多边形群）

cvFindChessboardCorners2

1. 自适应阈值分割（cvAdaptiveThreshold）
2. 对二值化后的图进行掩膜操作
3. 处理光照问题（reflection===1）
4. 循环体（形态学操作、找多边形、找数量为2的多边形群group2）
5. 对group2进行位置大小等条件的限制滤除，保留剩余的group2，并输出
6. 优化器

EOL\_str\_avm\_landmark数据包含检测角点的图像坐标和世界坐标,输入LM优化器，最后会得到相机模型的角度和位移四组六个参数；