MicroPos 定位系统编程手册

Developer's Guide

Made By Shuangyi

修订记录

Rev	Data	Author	Description
1.0	20160701	Shuangyi	创建初始版本
2.0	20170601	Shuangyi	
2.1	20171101	Shuangyi	
2.2	20200714	Shuangyi	更新部分函数

MicroPos 对位系统编程手册

第 1 章 开发说明

1.1 开发环境配置

在Windows系统下，用户可以使用支持动态链接库的开发工具来开发应用程序。目前定位模块只支持Visual Studio 2010开发环境。

使用Visual Studio 2010来开发定位应用流程：

【1】启动Visual Studio 2010， 新建一个工程；

【2】将安装目录的Library文件夹内dll文件、h文件和lib文件复制到工程文件夹中；

【3】选择“Project” 菜单下的“Properties…” 菜单项；

【4】切换到“Link” 标签页， 在“Object/library modules” 栏中输入lib 文件名MicroPos.lib；

【5】在应用程序文件中加入函数库头文件的声明，#include “MicroPos.h”

至此，用户就可以在Visual C＋＋中调用库中的函数开发定位应用程序。

将SYdef.h、MicroPos.h、MicroPos.lib、MicroPos.dll四文件放至开发工程目录之下，方可调用DLL文件中函数。

【注意】请将hast_rt.exe放在应用程序exe同一目录下，否则会报0XC0000044错误。

1.2 如何调用 MicroPos 相关函数

配置工作准备好后，接下来可以在 cpp 文件中调用函数了，方法如下所示：

1. **首先定义函数指针变量如下**

```
MicroPos *p_MicroPos = NULL;
```

2. **创建新的函数实体**

```
p_MicroPos = new CMicroPos(“Name”);
```

Name 是定位模块的名字，在多工位的系统中便于管理。我们以类成员函数的形式调用库函数，函数详细说明见第 2 章《函数说明》所示。

3. **结束时，删除函数指针变量**

```
if (p_MicroPos!= NULL)
{
    delete p_MicroPos;
    p_MicroPos = NULL;
}
```

在程序退出的时候，将建立的指针变量删除，将占用的内存资源释放。

第 2 章 函数说明

函数一览表	
1. SY_MP_GetVersion	获取版本信息
2. SY_MP_FitCircle	以圆周点估计圆心
3. SY_MP_RotateCenter_InPixel	3 点估计圆心
4. SY_MP_CalRotatePos	计算点旋转后新坐标
5. CalibrationSmallCameraFOV	小视野 3 点标定
6. CalculateSmallCameraFOVOffSet	小视野计算点到视野中心偏差量
7. SY_MP_XY_Calibrate	小视野 5 点标定
8. SY_MP_XY_Position	小视野定位函数
9. SY_MP_XYA_Calibrate	带标定板的标定函数
10. SY_MP_XYA_Free_Calibrate	自由标定函数
11. SY_MP_Sensor_To_World	映射函数（像素坐标转物理坐标）
12. SY_MP_World_To_Sensor	映射函数（物理坐标转像素坐标）
13. SY_MP_XYA_Position	带角度的定位函数（先看后抓）
14. SY_MP_Save_Parameters	保存标定参数
15. SY_MP_Load_Parameters	加载标定参数
16. SY_MP_Sort_Matrix	点阵排序函数
17. SY_MP_Create_Matrix	点阵生成函数
18. SY_MP_XYA_Position2	带角度的偏差计算函数（先抓后看）
19. SY_MP_Matrix_From_Model	点阵转换函数

2.1 动态链接库版本

```
char*       SY_MP_GetVersion()
```

返回当前定位模块 DLL 的版本信息

2.2 旋转轴标定

无论是大视野抓取和小视野定位的应用，只要涉及角度控制，都需要估计机械旋转中心在像素坐标系中的坐标。

2.2.1 看到整个圆周时使用此函数

```
Point2dF      SY_MP_FitCircle(int Count, Point2dF *Array)
```

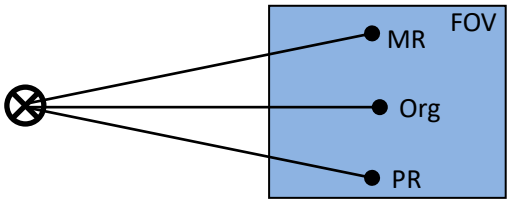
该函数返回拟合的圆心的坐标。

Count：用于估算旋转中心的视觉点数量

Array：Point2dF 类型数组，该数组保存图像处理得到的圆周运动视觉点坐标。

2.2.2 看到小于半个圆周时使用此函数

```
Point2dF      SY_MP_RotateCenter_InPixel(
                                    float Angle,
                                    Point2dF Org,
                                    Point2dF PR,
                                    Point2dF MR,
                                    bool b_CW
                                    );
```



Angle： 做顺时针或逆时针运动时的角位移量(degree)。

Org： Mark 点位于视野中心位置时的坐标。

PR： 顺时针旋转的像素坐标。

MR： 逆时针旋转的像素坐标。

b_CW： 正角位移为顺时针时为“TURE”，反之为“FLASH”，如上图所示的情况为“TRUE”。

2.3 计算旋转角度后新坐标

```
Point2dF      SY_MP_CalRotatePos(Point2dF RotateCeter,Point2dF MovePoint,float Angel)
```

通过该函数，计算 MovePoint 围绕旋转中心 RotateCeter 旋转 Angel 角度后的像素位置。

RotateCeter: 旋转中心的坐标位置。

MovePoint: 需要移动的坐标位置。

Angel: 需要旋转的角度(degree)。

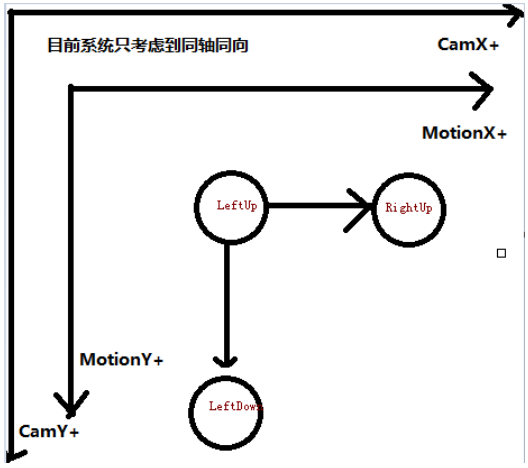
2.4 小视野 3 点标定与定位【不考虑角度】

2.4.1 标定函数

```
CamCalRes     CalibrationSmallCameraFOV(
                                        Point2dF LeftUpCamPoint,
                                        Point2dF RightUpCamPoint,
                                        Point2dF LeftDownCamPoint,
                                        Point2dF LeftUpAxisPoint,
                                        Point2dF RightUpAxisPoint,
                                        Point2dF LeftDownAxisPoint
                                        )
```

目前该函数只支持同轴同向的情况，通过 3 点标定 Camera 和 Plate 之间的关系，函数返回标定信息。详细信息见 CamCalRes。

下图所示为标定方式：



LeftUpCamPoint: 左上相机坐标

RightUpCamPoint:　　右上相机坐标

LeftDownCamPoint:　　左下相机坐标

LeftUpAxisPoint:　　左上轴位置

RightUpAxisPoint:　　右上轴位置

LeftDownAxisPoint:　　左下轴位置

2.4.2 定位函数

```
Point2dF      CalculateSmallCameraFOVOffSet(
                                            Point2dF MarkPoint,
                                            CamCalRes Calresult,
                                            int   ImgWidth,
                                            int   ImgHeight
                                            )
```

计算像素坐标 MarkPoint 至视野中心的物理偏移量。

MarkPoint:　　当前Mark点的位置

Calresult:　　小视野相机标定结果，通过函数CalibrationSmallCameraFOV获得。

ImgWidth:　　图像长度

ImgHeight:　　图像宽度

2.5 小视野 5 点标定与定位【不考虑角度】

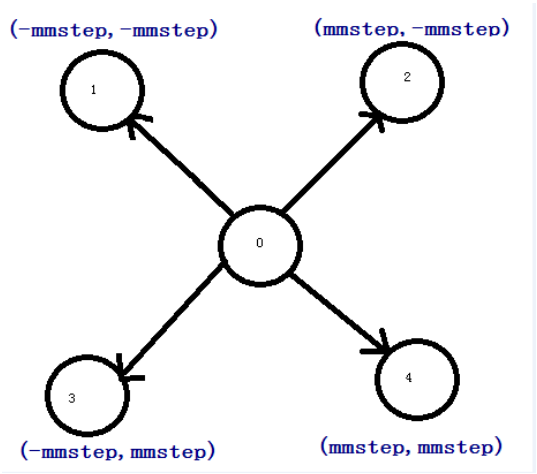
2.5.1 标定函数

```
PosParam      SY_MP_XY_Calibrate(Point2dF LandMark[5], float mmstep)
```

此函数主要针对 视野在 10mm*10mm 左右的定位应用 ，目前主要应用在打孔机。

LandMark[5]:　　视野中，按照mmstep步距，通过XY移动冲出的孔的像素坐标，详细见下图

mmstep:　　XY 轴移动步距



2.5.2 定位函数

```
PosData      SY_MP_XY_Position(
                              Point2dF SAMP,
                              Point2dF LandMark[5],
                              PosParam para,
                              float mmStep
                              )
```

该函数根据 5 点标定的信息，返回当前 Sample 位置所对应的 XY 位置偏移量。

SAMP:　　图像中的Mark点像素坐标。

LandMark[5]:用于五点标定中的5个像素坐标。

para:　　5点标定返回结果，通过函数SY_MP_XY_Calibrate获得。

mmStep:　　标定步距。

2.6 大视野标定与定位

标定函数 A 和 B，都用作标定相机坐标系和机械坐标系之间的关系。

前者针对带有标定板的标定方式；后者支持自由的像素与物理点对。

2.6.1 标定函数 A

```
double       SY_MP_XYA_Calibrate(
                              int ChessOrCircleGrid,
                              unsigned char *ChessGray,
                              int ImgWidth, int ImgHeight,
                              Point2dF LeftUp, Point2dF RightUp,
                              Point2dF LeftDown, Point2dF RightDown,
                              UINT cols, UINT rows,
                              Point2dF *MarkPoint,
                              CalibrationParameter *CP
                              )
```

带标定板的标定函数，返回标定误差。支持点阵标定板和棋盘格标定板。

ChessOrCircleGrid: 标定板的类型，其中0表示棋盘格标定板，1代表圆形点阵标定板

ChessGray:　　用于标定的灰度图像

ImgWidth:　　图像宽度

ImgHeight:　　图像高度

LeftUp:　　视野左上点的机械坐标系位置【通过试教点位得到，下同】

RightUp:　　视野右上点机械坐标系位置

LeftDown:　　视野左下点机械坐标系位置

RightDown:　　视野右下点机械坐标系位置

cols:　　点阵的列数

rows: 点阵的行数

MarkPoint: 点阵中的点坐标将保存在此数组，容量为cols rows

*CP: 标定信息

2.6.2 标定函数 B

```
double      SY_MP_XYA_Free_Calibrate(
                                    Point2dF *VP,
                                    Point2dF *MP,
                                    int ImgWidth,
                                    int ImgHeight,
                                    int TotalPointNum,
                                    CalibrationParameter *CP
                                    )
```

函数用户提供视觉与物理点对，完成标定并返回标定误差。

*VP: 视觉坐标系中的坐标

*MP: 机械坐标系中的坐标

ImgWidth: 图像长度

ImgHeight: 图像高度

TotalPointNum: 像素点与物理点的点对数量

*CP: 标定信息。

2.6.3 像素坐标与机器手物理坐标转换【不考虑角度】

像素坐标映射到物理坐标：

```
void    SY_MP_Sensor_To_World(
                            Point2dF *SensorPoint,
                            Point2dF *WorldPoint,
                            CalibrationParameter *CP
                            );
```

物理坐标映射到像素坐标：

```
void    SY_MP_World_To_Sensor(
                            Point2dF *WorldPoint,
                            Point2dF *SensorPoint,
                            CalibrationParameter *CP
                            );
```

*SensorPoint: 视觉坐标

*WorldPoint: 机械物理坐标

*CP: 标定信息，由标定函数SY_MP_XYA_Calibrate或SY_MP_XYA_Free_Calibrate得到。

2.7 大视野带角度的定位函数【“先看后抓”】

```
bool        SY_MP_XYA_Position(
                                Point2dF   CalRAngleXYPos,
                                float      RotateAngle,
                                Point2dF *FitCircleCenter,
                                Point2dF *SensorPoint,
                                Point2dF *WorldPoint,
                                CalibrationParameter *CP
                                )
```

大视野带角度定位抓取，适合“先看后抓”的工作方式。

CalRAngleXYPos：　估算旋转中心时机器手位置

RotateAngle：　产品角度【即机械手旋转角度】

*FitCircleCenter：　估算的旋转中心像素坐标

*SensorPoint：　产品像素坐标

*WorldPoint：　机械手的抓取 XY 位置

*CP：　标定信息，由标定函数SY_MP_XYA_Calibrate或SY_MP_XYA_Free_Calibrate得到。

2.8 保存系统参数函数

```
void      SY_MP_Save_Parameters(const char* Path, CalibrationParameter *CP);
```

该函数保存 MicroPos 中的相关标定结果。

Path：参数保存路径，一般选择系统文件保存的参数 ini 位置。

*CP：　标定信息，由标定函数SY_MP_XYA_Calibrate或SY_MP_XYA_Free_Calibrate得到。

2.9 加载系统参数函数

```
int       SY_MP_Load_Parameters(const char* Path, CalibrationParameter *CP);
```

该函数加载 MicroPos 中的相关标定结果。

Path：参数保存路径，一般选择系统文件保存的参数 ini 位置。

*CP：　标定信息，由标定函数SY_MP_XYA_Calibrate或SY_MP_XYA_Free_Calibrate得到。

2.10 点阵排序函数

```
void      SY_MP_Sort_Matrix(int Rows, int Cols, Point2dF *Points)
```

将Points数组提供的点阵按照X从左到右，Y从上到下的顺序排列，并更新到Points数组中。

Rows:　点阵的行数

Cols:　点阵的列数

*Points: 点阵数据

2.11 点阵生成函数

```
void      SY_MP_Create_Matrix(
                              int Rows, int Cols,
                              Point2dF LeftUp, Point2dF RightUp,
                              Point2dF LeftDown, Point2dF RightDown,
                              Point2dF *Points
                              )
```

给出点阵的左上、右上、左下和右下4点，即可根据设定的行列数生成点阵，以数组Points返回。

Rows:　　点阵行数
Cols:　　点阵列数
LeftUp:　点阵左上角坐标
RightUp:　点阵右上角坐标
LeftDown:　点阵左下角坐标
RightDown: 点阵右下角坐标
*Points:　点阵数组

2.12 带旋转的偏移计算函数【先抓后看】

```
Point2dF      SY_MP_XYA_Position2(
                                  Point2dF Org, Point2dF Samp,
                                  float DeltaAngle,
                                  Point2dF CCInPixel,
                                  CalibrationParameter *CP
                                  )
```

物料被拾取进入视野后，以Org作为目标位置计算带角度纠正的偏移量。

Org:　　目标位置像素坐标
Samp:　　样本点像素坐标
DeltaAngle: 样本角度与目标角度之差
CCInPixel:　在工作点位处机械旋转中心的像素坐标
*CP:　　标定信息，由标定函数SY_MP_XYA_Calibrate或SY_MP_XYA_Free_Calibrate得到。

2.13 点阵转换函数

```
void          SY_MP_Matrix_From_Model(
                                      Point2dF MP1, Point2dF MP2,
                                      Point2dF *Model,
                                      Point2dF NewMP1, Point2dF NewMP2,
                                      Point2dF *points,
                                      int NbPoints
                                      )
```

根据新的标记点，从Model数组转换生成新的点阵数组points。

MP1: 学习Model数组时，第1个主标记点坐标
MP2: 学习Model数组时，第2个主标记点坐标
*Model: 标准的模板点阵数组
NewMP1: 新的第1个主标记点坐标
NewMP2: 新的第2个主标记点坐标
*points: 转换后点阵数组
NbPoints: Model模板点阵数据的个数【不包含2个主标记点】

第 3 章 常用结构体说明

3.1 小视野三点标定结果结构体

```
typedef struct CamCalibrationRes
{
    float         m_fCamXDeg;       //相机 X 方向与轴的夹角 deg
    float         m_fCamYDeg;       //相机 Y 方向与轴的夹角 deg
    float         m_fCamXReslotion; //相机 X 方向的解析度   mm/pix
    float         m_fCamYReslotion; //相机 Y 方向的解析度   mm/pix
}CamCalRes;
```

3.2 五点标定结果结构体

```
typedef struct PosParam
{
    float V1;
    float V2;
    float V3;
    float V4;
}PosParam;
```

3.3 偏移量结构体

```
typedef struct PosData
{
    float xmm;        //mm
    float ymm;        //mm
    float angle;      //deg
}PosData;
```