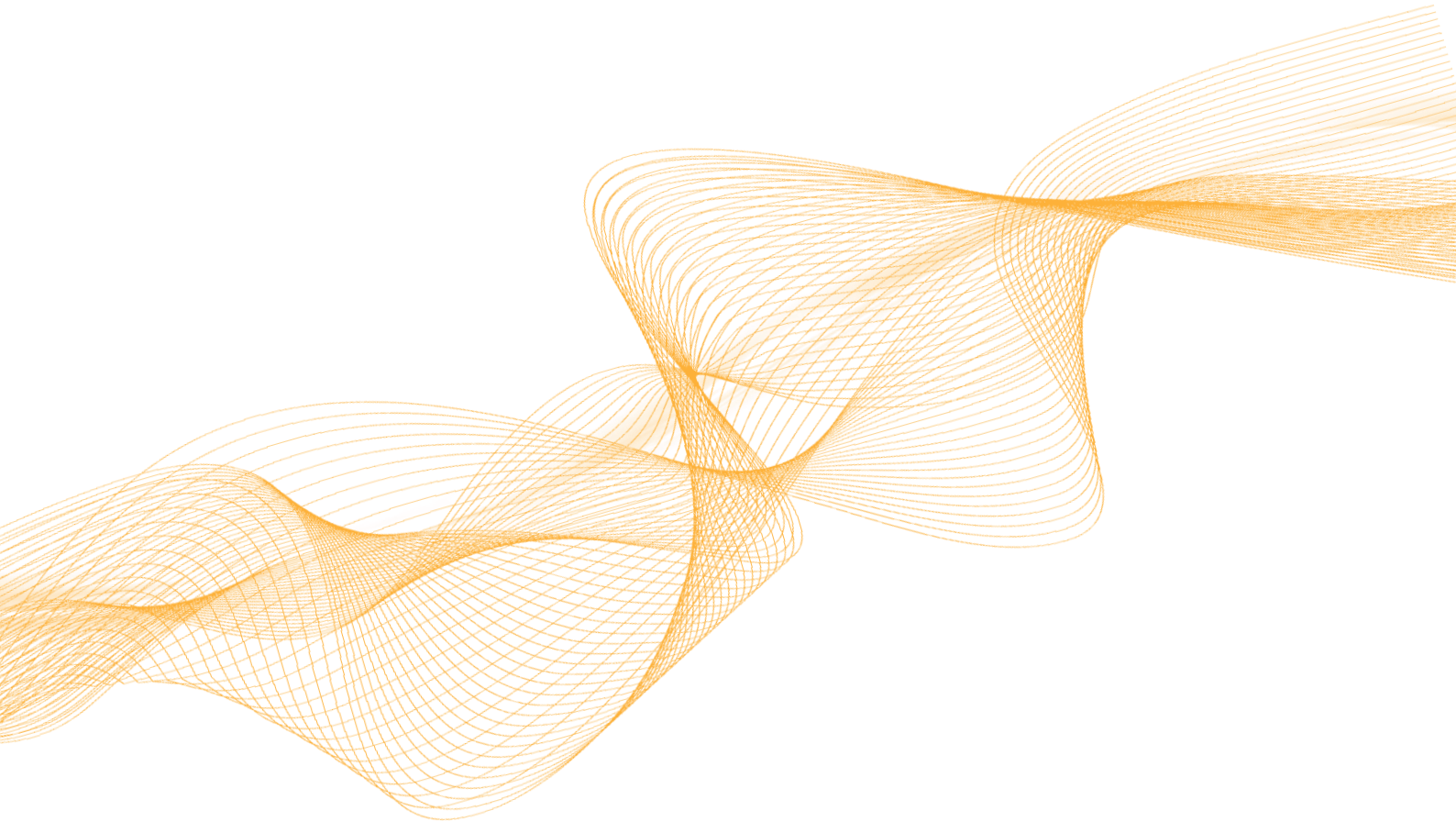
双翌
SHUANGYI

MicroPos 开发手册



深圳市双翌光电科技有限公司

修订记录:

Rev	Data	Author	Description
1.0	20160701	Shuangyi	创建初始版本
2.0	20170601	Shuangyi	新增阵列函数接口
2.1	20171101	Shuangyi	新增获取两坐标系间的角度函数接口
2.2	20200714	Shuangyi	更新部分函数
3.0	20230110	Shuangyi	更新部分函数

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

目　　录

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

简介

MicroPos 机械手视觉定位算法是双翌专门针对机械手在高精度应用场合需求而自主研发的软件产品算法包，MicroPos 机械手视觉定位算法助您快速构建机械手智能装配、机械手上下料、机械手精密组装、机械手贴合等等行业应用。算法应用广泛、具有强大的兼容性，使得工程人员有更多时间和精力来开发自己的产品。

MicroPos 机械手视觉定位算法应用领域：

- 机械手大视野抓取产品应用领域
- 机械手视觉摆料应用
- 机械手定位贴标应用
- 机械手视觉定位打螺丝应用
- 机械手视觉点胶应用
- 机械手智能装配

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

第 1 章 MicroPos 开发说明

软件支持函数库

MicroPos 机械手视觉定位算法支持 Windows7、Windows10 等 32/64 位操作系统，并提供完整的函数库与动态链接库(DLL)，用户可以轻松完成其应用程序。

支持 IDE(Integrated Development Environment,集成开发环境)

开发语言	Using……	操作系统 (OS)	开发环境	版本	MicroPos 版本
C++	C++ classes	Windows 7	MS Visual Studio C++	15.0	支持

编码:Unicode 编码方式

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_CopyFrom	拷贝其它对象数据直接使用
3.SY_MP_FitCircle	以圆周点估计圆心，圆周点大于 3 点可使用
4.SY_MP_RotateCenter_InPixel	3 点估计圆心
5.SY_MP_CalRotatePos	计算点旋转后新坐标
6.CalibrationSmallCameraFOV	小视野 3 点标定
7.CalculateSmallCameraFOVOffSet	小视野计算点到视野中心偏差量
8.SY_MP_XY_Calibrate	小视野 5 点标定
9.SY_MP_XY_Position	小视野定位函数
10.SY_MP_XYA_Calibrate	带标定板的标定函数
11.SY_MP_Check_Calibration	检查标定板方法是否标定正确
12.SY_MP_XYA_Free_Calibrate	自由标定函数
13.SY_MP_XYA_Free_Calibrate_M	自由标定函数，自动分区域标定
16.SY_MP_Find_MaxError_M	自动分区域标定应用中，计算最大映射误差函数
14.SY_MP_Sensor_To_World	映射函数（像素坐标转物理坐标）
15.SY_MP_World_To_Sensor	映射函数（物理坐标转像素坐标）
20.SY_MP_XYA_Position	带角度的定位函数（先看后抓）
21.SY_MP_Save_Parameters	保存标定参数
22.SY_MP_Load_Parameters	加载标定参数
23.SY_MP_Sort_Matrix	点阵排序函数
24.SY_MP_Create_Matrix	点阵生成函数
25.SY_MP_XYA_Position2	带角度的偏差计算函数（先抓后看）
27.SY_MP_Matrix_From_Model	点阵转换函数
28.SY_MP_See_To_Hit	只标定一次，移动任意位置观察目标，计算偏移量
29.SY_MP_See_To_Hit2	带前端工具坐标的 SY_MP_See_To_Hit 功能
30.SY_MP_Distance	计算两独立坐标系中点的物理距离
31.SY_MP_Calibrate_Tool	标定前端工具坐标
32.SY_MP_Undistort_Image	纠正图像畸变
33.SY_MP_Perspctive_Train	透视变换训练
34.SY_MP_Perspctive_Transform	透视变换
35.SY_MP_Invert_Perspctive_Transform	透视逆变换

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 对象拷贝

```
void       SY_MP_CopyFrom(MicroPos *SrcPos)
```

从其它对象拷贝全部信息至本对象

2.3 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。

2.3.1 看到整个圆周时使用此函数

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

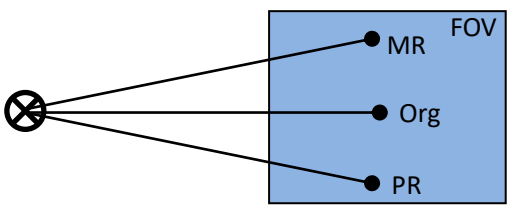
该函数返回拟合的圆心的坐标。

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。

2.3.2 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”。

2.4 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)。

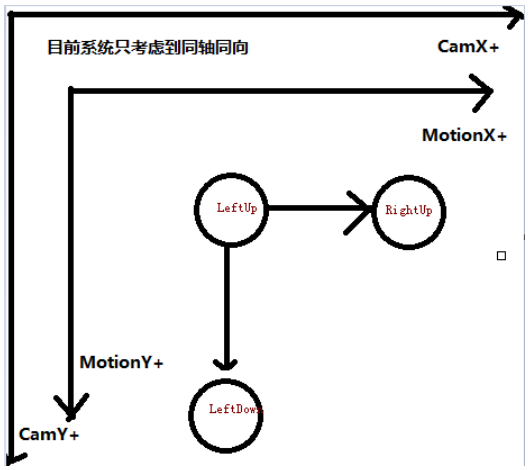
2.5 小视野 3 点标定与定位【不考虑角度】

2.5.1 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

2.5.2 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

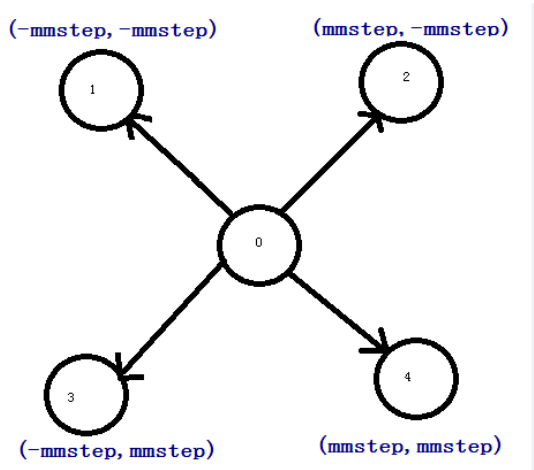
2.6 小视野 5 点标定与定位【不考虑角度】

2.6.1 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



2.6.2 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。

2.7 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。

2.7.1 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,
                              unsigned int cols, unsigned int rows,
                              Point2dF *MarkPoint
                              )
```

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

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

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

2.7.2 标定结果检查

```
void        SY_MP_Check_Calibration(HDC dc,
                                    Point2dF *MarkPoint, int NumOfMarkPoint,
                                    float fx, float fy)
```

用可视化的方式检查 **标定函数A** 的标定结果，被分析到的靶标将会从上到下，从左到右用直线连接。
dc:　　MFC中的设备上下文HDC
MarkPoint:　　由标定函数A分析到的靶标序列
NumOfMarkPoint:　　靶标序列中的元素个数
fx、fy:　　绘画至HDC时的比例

2.7.3 标定函数 B

```
double      SY_MP_XYA_Free_Calibrate(
                                    Point2dF *VP,
                                    Point2dF *MP,
                                    int ImgWidth,
                                    int ImgHeight,
                                    int TotalPointNum,
                                      Calibration_Mode mode
                                    )
```

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
mode:　　标定模型（FishEye、 HandEye）

2.7.3 标定函数 C　自动分区域标定及误差检查函数

```
double      SY_MP_XYA_Free_Calibrate_M(
                                        Point2dF *VP, Point2dF *MP,
                                        int ImgWidth, int ImgHeight,
                                        int rows, int cols, Calibration_Mode mode)
```

自动根据行列数，进行分区域标定，以提高大视野标定的整体精度，特别是在视野边缘的精度。

*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
rows:　　标定矩阵行数
cols:　　标定矩阵列数
mode:　　标定模型（FishEye、 HandEye）

```
void        SY_MP_Find_MaxError_M(
                                    Point2dF *VP, Point2dF *MP,
                                    int rows, int cols,
                                    int &MaxIdx, float &MaxError)
```

自动分块标定后，找出VP与MP互映射的最大误差及对应的点对下标
*VP:　　视觉坐标系中的坐标
*MP:　　机械坐标系中的坐标
rows:　　标定矩阵行数
cols:　　标定矩阵列数
MaxIdx:　　最大误差的点对下标
MaxError:　　最大误差值

2.7.4 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
                            )
```

物理坐标映射到像素坐标：

```
void     SY_MP_World_To_Sensor(
                            Point2dF *WorldPoint,
                            Point2dF *SensorPoint
                            )
```

*SensorPoint:　视觉坐标
*WorldPoint:　机械物理坐标

2.8 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
                    )
```

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

2.9 参数保存与加载

```
void    SY_MP_Save_Parameters(const char* Path)
```

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。

```
int     SY_MP_Load_Parameters(const char* Path)
```

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。

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

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

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
                               )
```

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

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】

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

2.14 任意位置定位函数

```
void        SY_MP_See_To_Hit(
                        Point2dF PosOfCal, float J1AngleOfCal, float J2AngleOfCal,
                        Point2dF *SeePoint, Point2dF PosOfSee,
                        float J1AngleOfSee, float J2AngleOfSee, Point2dF *HitPos)
```

四轴机械手应用，工作点在 Z 轴中心。只标定一次，移动至任意位置观察目标并计算出相对偏移量，对目标进行定位。

PosOfCal:　　标定时机械手绝对位置
J1AngleOfCal:　　标定时 J1 关节角度
J2AngleOfCal:　　标定时 J2 关节角度
SeePoint:　　观察的靶标像素坐标
PosOfSee:　　观察靶标时机械手的绝对位置
J1AngleOfSee:　　观察靶标时 J1 关节角度
J2AngleOfSee:　　观察靶标时 J2 关节角度
HitPos:　　靶标的绝对位置

```
void        SY_MP_See_To_Hit2(
                        Point2dF ToolPos, Point2dF PosOfCal,
                        float J1AngleOfCal, float J2AngleOfCal,
                        Point2dF *SeePoint, Point2dF PosOfSee,
                        float J1AngleOfSee, float J2AngleOfSee, Point2dF *HitPos)
```

ToolPos：前端工具坐标，使用 SY_MP_Calibrate_Tool（）标定得到

2.15 距离函数

```
float           SY_MP_Distance(Point2dF CCA, Point2dF A, Point2dF CCB, Point2dF B)
```

计算两物理点的直线距离。

CCA、CCB:　　两局部坐标中标定的物理旋转中心
A 、B:　　两局部坐标中的物理点

2.16 标定前端工具

```
Point2dF        SY_MP_Calibrate_Tool(
                    Point2dF RobotPosA, Point2dF RobotPosB,    float RotateAngle)
```

标定前端工具坐标。

RobotPosA:　　旋转前端工具对准靶标的机械手位置
RobotPosB:　　旋转 RotateAngle 后前端工具对准靶标的机械手位置
RotateAngle:　　旋转的角度

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

2.17 图像畸变纠正

```
void        SY_MP_Undistort_Image(
                                unsigned char *pSrc, int SrcWidth, int SrcHeight,
                                unsigned char *pDst, int DstWidth, int DstHeight)
```

利用标定信息，纠正图像畸变，并输出被纠正后的图像。

pSrc:　　源图像的首地址
SrcWidth:　源图像的宽度
SrcHeight:　源图像的高度
pDst:　　目标图像的首地址
DstWidth:　目标图像的宽度
DstHeight:　目标图像的高度

2.18 透视变换训练

```
void            SY_MP_Perspctive_Train(Point2dF src[4], Point2dF dst[4])
```

利用 4 个坐标点对进行透视训练，得到变换矩阵。

src[4]：原空间中的 4 个坐标
dst[4]：目标空间中，对应 src[4]的 4 个坐标

2.19 透视变换

```
void        SY_MP_Perspctive_Transform(Point2dF *srcPoint, Point2dF *dstPoint)
```

坐标点透视变换。

srcPoint：原空间坐标点
dstPoint：目标空间坐标点

```
void        SY_MP_Perspctive_Transform(
                                unsigned char *pSrc, int SrcWidth, int SrcHeight,
                                unsigned char *pDst, int DstWidth, int DstHeight)
```

图像透视变换，把图像从源空间通过透视变换到目标空间，得到一张新的图像。

pSrc:　　原空间图像的首地址
SrcWidth:　原空间图像的宽度
SrcHeight:　原空间图像的高度
pDst:　　目标空间图像的首地址
DstWidth:　目标空间图像的宽度
DstHeight:　目标空间图像的高度

2.20 透视逆变换

```
void    SY_MP_Invert_Perspctive_Transform(Point2dF *dstPoint, Point2dF *srcPoint)
```

坐标点透视逆变换。

dstPoint：目标空间坐标点

srcPoint：原空间坐标点

```
void     SY_MP_Invert_Perspctive_Transform(
                                    unsigned char *pDst, int DstWidth, int DstHeight,
                                    unsigned char *pSrc, int SrcWidth, int SrcHeight)
```

图像透视逆变换，把图像从目标空间通过透视变换回到原空间。

pDst：　　目标空间图像的首地址

DstWidth：　目标空间图像的宽度

DstHeight：　目标空间图像的高度

pSrc：　　原空间图像的首地址

SrcWidth：　原空间图像的宽度

SrcHeight：　原空间图像的高度

地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

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;
```

3.4 标定模型

```
enum Calibration_Mode
{
    FishEye,        //带有镜头畸变因素的两空间标定模型
    HandEye         //不带镜头畸变因素的两空间标定模型
};
```


地址：深圳市宝安区沙井街道茅洲山工业园全至科创大厦2A-1
电话：0755-23712116
传真：0755-23020631

双翌
SHUANGYI

0755-23712116

网址：www.shuangyi-tech.com
邮箱：contact@shuangyi-tech.com
地址：深圳市宝安区沙井街道后亭茅洲山工业园全至科创大厦2A-1

微信公众号