XPos 光学定位技术指引

上海隶首信息技术有限公司

Ver1.0

2013-12-30

概述

在印刷品等切割行业，存在对印刷误差、切割物料摆放误差就行纠正的需求。目前通过印刷时印制的定位标识，在切割时对这些标识定位后，变换切割轮廓的解决方案，已经是行业内的通行做法。

XPos 定位模块提供了基于民用摄像头视觉定位的算法逻辑，定位精度在一个像素以内。（在 30 万像素摄像头—640*480、视野面积 50mm 宽的情况下，每个像素大约 0.078mm。）

民用普通摄像头和工业摄像头相比，主要是镜头变形大（广角），感光能力差，对识别算法要求较高，但是价格低到可以忽略，尤其是可以通过软件技术客服上述弱点的情况下，这个解决方案还是比较有吸引力的。

目前 XPos 只提供了摄像头和定位计算支持，对于驱动切割机运动，对定位后整个矢量图的旋转、平移、拉伸等变换没有提供接口支持。如果需要，我们可以提供额外的技术配合开发。

XPos 是 Windows 平台上的一组 DLL，通过软件锁保护。可以在 VC6 或更高版本的 VC 工程中调用。

XPos 具体的接口函数请阅读附录，开发接口.h 文件内容，并请参阅演示代码工程 XPosDemo。

基本说明

1. 应该在一个主窗口内创建并保持摄像头预览窗口。
2. 切割机安装或者使用一段时间之后，进行校正。
3. 在定位之前，需要先设定定位模版。
4. 调用软件系统需要根据任务，确定每个定位点的拍照位置（一般是至少 3 个），在切割机运动到每个位置后，调用定位计算函数，确定视野内的定位图形的切割机坐标。从而计算整个矢量图在 XY 平面中的旋转、平移和拉伸（这个计算由调用系统自己完成）。

关于定位校正

定位校正的目的是建立摄像头安装位置、角度误差和切割机坐标之间的变换。这里的误差不仅仅来源于摄像头和安装刀具支座之间的安装误差，还包括切割机横梁和轨道的机械变形误差——也就是说，横梁和轨道可能是弯曲和扭曲的，支座在其上运动到不同的位置，将对应不同的摄像头位置和角度误差。

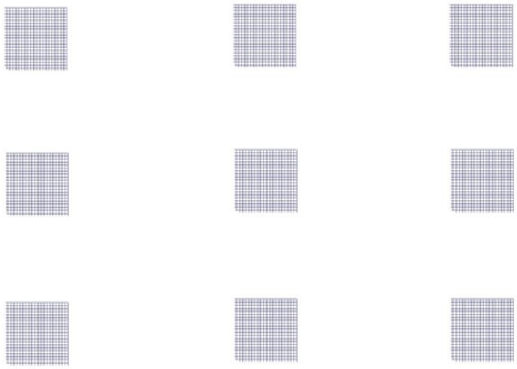
具体校正方法流程如下：

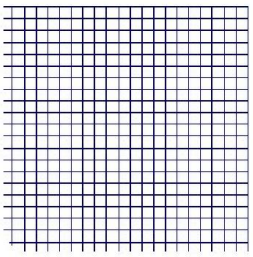
1. 根据切割机特性（机械精度、幅面等）参数，创建一个用于校正的网格组 PLT。

   a) 特性参数保存在配置 ini 文件中。

   b) XPos 提供了创建 PLT 文件的接口函数。

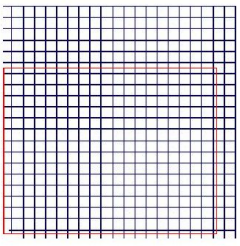
   c) 这个 PLT 文件可以保留应用于同型号切割机。





   d) 各组网格平均分布于切割机幅面区域，比如上图左示例 3*3 共 9 组。每组网格都是如上图右所示：缺省是 5mm 间距，总面积能够覆盖摄像头的视野。最左下角点是特殊的，称为【基点】。具体网格组分布通过 ini 文件配置定义，需要根据切割机幅面和机械精度分布特征确定。

2. 把上述 PLT 网格组用切割机的笔绘制在平铺的纸张之上，并保持校正过程中，纸张不动。

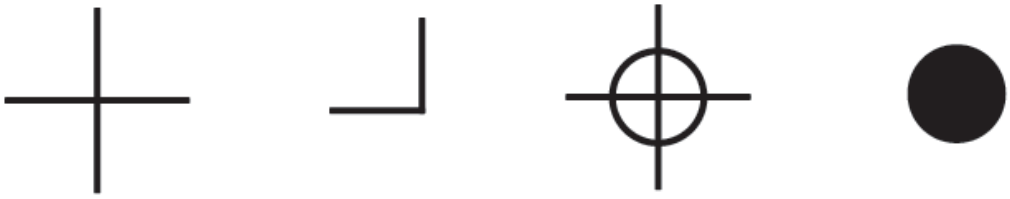


3. 控制切割机运动到最左下角的网格组，使得摄像头视野如右图所示，【基点】距离左下角大约 1/2 到 1/3 网格距离。然后调用校正接口函数，执行校正。校正过程需要对每个网格组执行一次拍照操作。

4. 校正成功后，所有信息都会保存在数据路径之中。

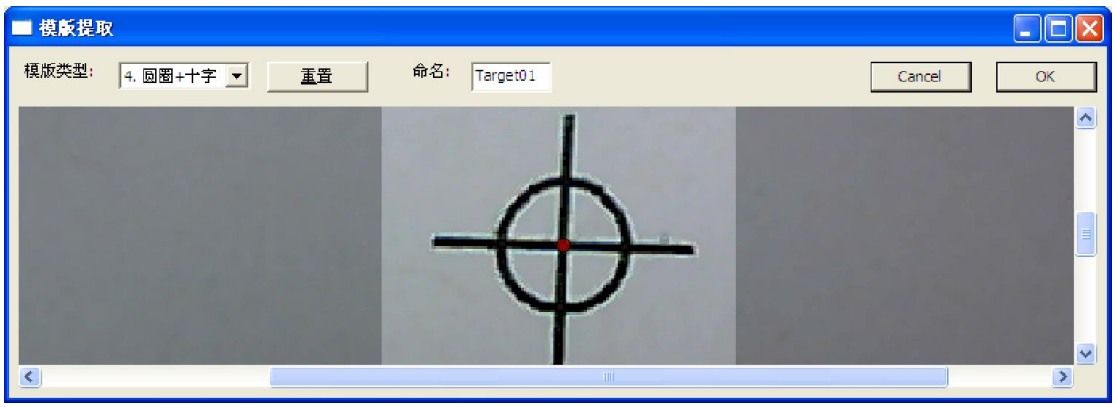
关于设定模版

目前 XPos 支持 5 种模版，其中前四种具体如下图所示：



第五种是任意图形匹配。

设定模版操作需要移动切割机到包含定位点图形的地方拍照，并在相应界面中框选图形作为匹配模版。操作中需要指定模版类型，对于前 4 中模版，框选完成后 XPos 即可自动识别模版的中心点。如果识别失败或有偏差，用户可以人工干预，修改中心点。第五种模版则必须人工设定中心点。



模版设定后保存在数据路径中。XPos 提供了列、取历史模版的函数。

定位

定位函数返回当前切割机视野中，和模版图案匹配的定位点之切割机坐标。定位函数提供有一个匹配度参数，可以设定匹配度的最低阀值，在匹配度低于该值时，弹出人工干预界面，手工指定定位点。

附录——XPosPub.h

```
#pragma once

#ifdef XPOS_EXPORTS
    #define XPOS_DLL_FUC   AFX_CLASS_EXPORT
#else
    #define XPOS_DLL_FUC   AFX_CLASS_IMPORT
```

```
    #pragma comment( lib, "xpos.lib" )
#endif

/*************************************
基本说明:
    1. 应该在一个主窗口内创建并保持摄像头预览窗口(B.1);
    2. 切割机安装后，进行校正; 校正参数保存在指定数据路径中的ini文件中。
        2.1 校正参数是一组网格单元组成，每个网格单元目前是固定的5mm间距的网格，大小比摄像头视野稍大（50*40）
        2.2 切割机的参数和校正参数可以通过(C.0.1), (C.0.2)和(C.0.3)设定;校正所用网格PLT文件由(C.1)创建, 安装参数不变则可以一直使用这个PLT文件
        2.3 校正网格PLT的基点定义为——最靠近左下角点的网格交叉点
        2.4 校正过程是在切割机上绘制好校正网格PLT(C.1)后，对各个网格依次拍照并执行校验的过程。
            首先使用(C.0.4 iIndex=-1)，获得设计参数计算出来的第一次拍照的切割机坐标，并执行校正(C.2 pParrent!=NULL);
            如果位置偏差较大，则取消并手工移动切割机到合适位置再次执行校正(C.2 pParrent!=NULL)
            成果后记录这个切割机坐标为ptBasePos，重复调用(C.0.4 iIndex=1,2,...) 直到返回XPosERR_POSOUT表示iIndex超出了范围结束
            每次获得ptTargetPos后，执行校正(C.2 pParrent==NULL)，只有在失败的时候才进行手工干预——再次执行校正(C.2 pParrent!=NULL);
    3. 定位运算时，先进行模版设定(D.1); 然后在每个取样点上进行定位计算(E)
*************************************/

////////////////////////////////////
//A. 基础定义
//A.1 返回值 --- 所有接口函数的返回值都是int表示的错误码
#define XPosERR_OK              0x0000    //成功

#define XPosERR_NOTINIT         0x0001    //没有初始化
#define XPosERR_BADPARAM    0x0002    //参数异常
#define XPosERR_CANCEL      0x0003    //用户取消
#define XPosERR_SYSTEM      0x0004    //系统错误
#define XPosERR_FILEIO      0x0005    //文件系统访问出错
#define XPosERR_NODONGLE    0x0006    //没有软件锁
#define XPosERR_MODULELOST 0x0007    //丢失软件模块

#define XPosERR_NOCAM       0x000A    //没有找到摄像头
#define XPosERR_NOTREADY    0x000B    //摄像头连接失败，系统异常或者是摄像头被占用
#define XPosERR_NOTSUPPORT 0x000C    //不支持的功能：比如参数设置对话框等
#define XPosERR_NOTADJUSTED     0x000D    //没有校正信息。定位之前，必须校正
#define XPosERR_NOPATTERN   0x000E    //没有设定模版图像, 或者读取历史模版失败
```

```
#define XPosERR_POSOUT      0x000F   //定位失败，识别出的定位点不在校正范围内
#define XPosERR_POSFAILED  0x0010   //定位失败，没有识别出定位点

//A.2 点数据定义
struct DoublePoint
{
    double x;
    double y;
    DoublePoint() {}
    DoublePoint(double _x, double _y) : x(_x), y(_y) {}
};


////////////////////////////////////
//B. 初始化和摄像头基本操作
//   所有操作之前必须调用初始化函数；
//   一般的摄像头窗口应该放在一个固定的父窗口之内，用的时候显示，不用的时候隐藏；初始化成功后，摄像头窗口是隐藏状态；
//   可以移动和修改该窗口的大小；
//   摄像头驱动提供了拍照参数调节和分辨率设定对话框。
//B.1 初始化软件狗，读取ini配置参数，创建摄像头窗口。
//参数：
//   【IN】const char* szDataPath 工作路径：保存ini，模版图像文件，调试情况下保存照片文件等
//   【IN】CWnd* pParrent 父窗口
//   【IN】int iPreviewSizeX, int iPreviewSizeY 创建摄像头窗口的大小（32和2048之间）
//   【IN】int iPosX, int iPoxY 创建窗口在父窗口之内的位置，左上角点坐标
XPOS_DLL_FUC int XPos_Init(const char* szDataPath, CWnd* pParrent, int iPreviewSizeX, int iPreviewSizeY, int iPosX=0, int iPoxY=0);

//B.2 重置——考虑USB线不稳定情况，重新连接摄像头
XPOS_DLL_FUC int XPos_Reconnect();

//B.3 显示/隐藏预览拍照窗口
//参数：
//   【IN】bool bShow 动作：true - 显示；false - 隐藏
//   【IN】int* piPosX=NULL, int* piPoxY=NULL 不是NULL时，可以把窗口位置移动
//   【IN】int* piPreviewSizeX=NULL, int* piPreviewSizeY=NULL 不是NULL时，可以修改窗口的大小
XPOS_DLL_FUC int XPos_ShowWindow(bool bShow, int* piPosX=NULL, int* piPoxY=NULL, int* piPreviewSizeX=NULL, int* piPreviewSizeY=NULL);

//B.4 显示摄像头驱动提供的参数调节对话框
```

```
//B.4.1 摄像头参数控制对话框
XPOS_DLL_FUC int XPos_DlgCtrl_DoModal();
//B.4.2 摄像头分辨率设定对话框
XPOS_DLL_FUC int XPos_DlgFmt_DoModal();


////////////////////////////////////////////////////////////////////////
//C. 校正
//-------------------------------------
//C.0 辅助函数，关于校正网格组的参数
//C.0.1 切割机校正范围
//参数：
//   【IN】const DoublePoint &ptMin 切割机的坐标范围最小值，缺省(0,0)，mm单位
//   【IN】const DoublePoint &ptMax 切割机的坐标范围最大值，缺省(1700,1300)，mm单位
XPOS_DLL_FUC int XPos_AdjFrame_SetRange(const DoublePoint &ptMin, const DoublePoint
&ptMax);

//C.0.2 切割机摄像头安装偏移参数
//参数：
//   【IN】const DoublePoint &ptCamOffset 切割机坐标相对于摄像头左下角偏移量，mm单位 缺省
值(20,20)
//       根据这个偏移值，可以在校正时，把摄像头移动到合理的位置，进行第一次定位尝试
XPOS_DLL_FUC int XPos_AdjFrame_SetCamOffset(const DoublePoint &ptCamOffset);

//C.0.3 网格组的分布数——在XY方向各分布几个网格，这个参数取决于误差分析
//参数：
//   【IN】const CPoint &ptFrameNum 在XY方向分别输出的网格数目；
//       根据这个数目和切割机的坐标范围，确定网格的分布——网格的间距等于最边上网格距离
边界的2倍
XPOS_DLL_FUC int XPos_AdjFrame_SetNumber(const CPoint &ptFrameNum);

//C.0.4 获取校正过程中需要执行拍照校正的坐标
//       首先使用iIndex=-1，获得设计参数计算出来的第一次拍照的切割机坐标，并执行校正(C.2
pParrent!=NULL)；
//       如果位置偏差较大，则取消并手工移动切割机到合适位置再次执行校正(C.2
pParrent!=NULL)
//       成果后记录这个切割机坐标为ptBasePos，重复调用iIndex=1,2,... 直到返回
XPosERR_POSOUT表示iIndex超出了范围结束
//       每次获得ptTargetPos后，执行校正(C.2 pParrent==NULL)，只有在失败的时候才进行手工
干预——再次执行校正(C.2 pParrent!=NULL)；
//参数：
//   【IN】const DoublePoint &ptBasePos 基点单元的拍照坐标 -- 切割机坐标; iIndex==-1忽略
//   【IN】int iIndex 第iIndex个网格单元的对应拍照坐标 (0时就是传入参数ptBasePos的坐标),
```

```
(-1时返回设计参数计算出来的估计坐标位置)
//   【OUT】DoublePoint &ptMoveToPos 目标单元的拍照坐标 -- 切割机坐标
XPOS_DLL_FUC int XPos_AdjFrame_GetTargetPos(const DoublePoint &ptBasePos, int iIndex,
DoublePoint &ptTargetPos);


//---------------------------------------
//C.1 辅助函数，创建校正网格PLT文件。正常情况下，PLT文件保存一个固定的就可以了
//   【IN】const char* scszPathfilename PLT文件名；NULL则保存到初始化提供的数据路径中
frame.plt
XPOS_DLL_FUC int XPos_AdjFrame_CreatePLT(const char* scszPathfilename);

//---------------------------------------
//C.2 拍照并显示校正对话框——拍摄切割机绘制的、C.1输出的网格；
//说明：
//   只有指定pParrent时，这个函数才会显示对话框。只对最靠近原点的网格单元拍摄和自动计算失
败需要人工干预时，才需要显示对话框确认合理的拍照位置。
//   合理视野范围：注意把目标网格的左下角点基点包括在内，并且基点距离视野边界1/3~1/2格的
距离
//   其他的网格单元应该传入参数pParrent=NULL，自动计算不需要显示校正对话框。
//参数：
//   【IN】const DoublePoint &ptCurrentPos 当前切割机坐标位置
//   【IN】CWnd* pParrent  父窗口；pParrent!=NULL显示校正对话框，否则不显示对话框
XPOS_DLL_FUC int XPos_AdjFrame_CatchPhoto(const DoublePoint &ptCurrentPos, CWnd*
pParrent);


////////////////////////////////////////////////////////////////////////
//D. 定位模版
//   模版支持4种特定类型和1个一般类型；特定类型的模版、定位精度要高。
//   指定特定类型的模版并在模版图中用鼠标拉框选择模版后，系统自动识别模版的中心位置；失
败则只能按照一般类型PatternType_IMGSHAPE执行。
//   模版成功设定后，信息将保存在初始化给定的数据路径中，其中模版图保存在子目录Pattern
中。
//D.1 创建并设定当前定位模版——拍照并显示提取模版对话框
//参数：
//   【IN】CWnd* pParrent 父窗口
//   【OUT】char* szPatternName 新创建模版的名字, 名字最大允许[64]
XPOS_DLL_FUC int XPos_Pattern_New(CWnd* pParrent, char* szPatternName);

//D.2 查询当前定位模版类型和名字
//参数：
//   【OUT】char* szPatternName 模版的名字, 名字最大允许[64]
```

```
XPOS_DLL_FUC int XPos_Pattern_GetCurrent(char* szPatternName);

//D.3 查询历史模版总数
//参数:
//   【OUT】int& iPatternCount 返回的模版总数
XPOS_DLL_FUC int XPos_Pattern_GetCount(int& iPatternCount);

//D.4 查询第iPattern个历史模版的名字
//参数:
//   【IN】int iPattern 要返回的模版序号, 范围在0~XPos_Pattern_GetCount之间
//   【OUT】char* szPatternName 模版的名字, 名字最大允许[64]
XPOS_DLL_FUC int XPos_Pattern_GetName(int iPattern, char* szPatternName);

//D.5 删除指定历史模版
//参数:
//   【IN】const char* szPatternName 模版的名字
XPOS_DLL_FUC int XPos_Pattern_Delete(const char* szPatternName);

//D.6 设定历史模版为当前模版
//参数:
//   【IN】const char* szPatternName 模版的名字
XPOS_DLL_FUC int XPos_Pattern_SetHistory(const char* szPatternName);

//D.7 编辑查看历史模版——显示提取模版对话框，加载历史模版数据
//参数:
//   【IN】CWnd* pParrent 父窗口
//   【IN】const char* szPatternName 模版的名字
XPOS_DLL_FUC int XPos_Pattern_Edit(CWnd* pParrent, const char* szPatternName);


////////////////////////////////////
//E. 定位函数: 执行一次拍照并定位计算
//   如果需要多尝试一次拍照定位(光照等条件影响导致识别失败情况可能不做任何移动，再次拍照计算就可能成功)，
//   则第一次应该指定fMatchVal=0绝对不启动对话框人工干预, 然后根据返回的匹配度值判定是否成功，还是进行再一次的尝试。
//参数:
//   【OUT】DoublePoint& ptResult 返回的切割机坐标位置
//   【INOUT】float& fMatchVal 匹配度，传入参数影响手工定位对话框启动设定：<=0 不启动对话框; >=1 无条件启动对话框; (0~1) 当实际定位匹配度低于此参数时才弹出对话框
//            函数返回XPosERR_OK or XPosERR_POSFAILED: 返回定位匹配度
//   【IN】const DoublePoint &ptCurrentPos 当前切割机坐标位置
//   【IN】CWnd* pParrent  父窗口
```

```
XPOS_DLL_FUC int XPos_FindPosition(DoublePoint& ptResult, float& fMatchVal, const
DoublePoint &ptCurrentPos, CWnd* pParrent);
```