电压表示数识别 API 使用手册

一、环境配置

1. 新建项目文件。
2. 打开属性管理器，找到 Debug|x64，右击选择属性，如图 1.1 所示。选择 VC++目录，在包含目录中添加路径 1: \opencv3.4.1(最全最详细配置)\opencv\build\include，路径 2: \opencv3.4.1(最全最详细配置)\opencv\build\include\opencv2 。如图 1.2 所示，单击确认键。

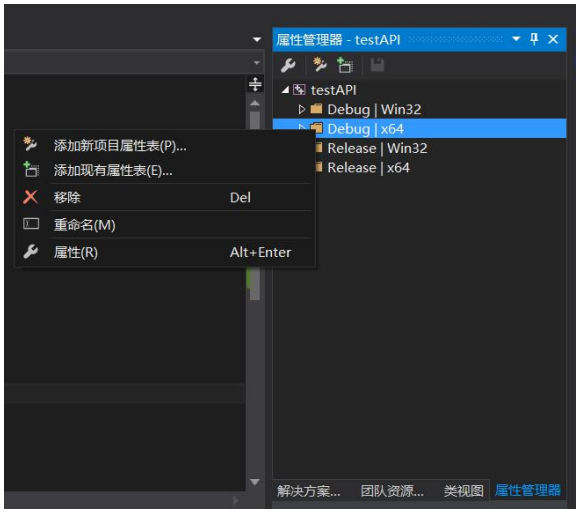

图 1.1 配置

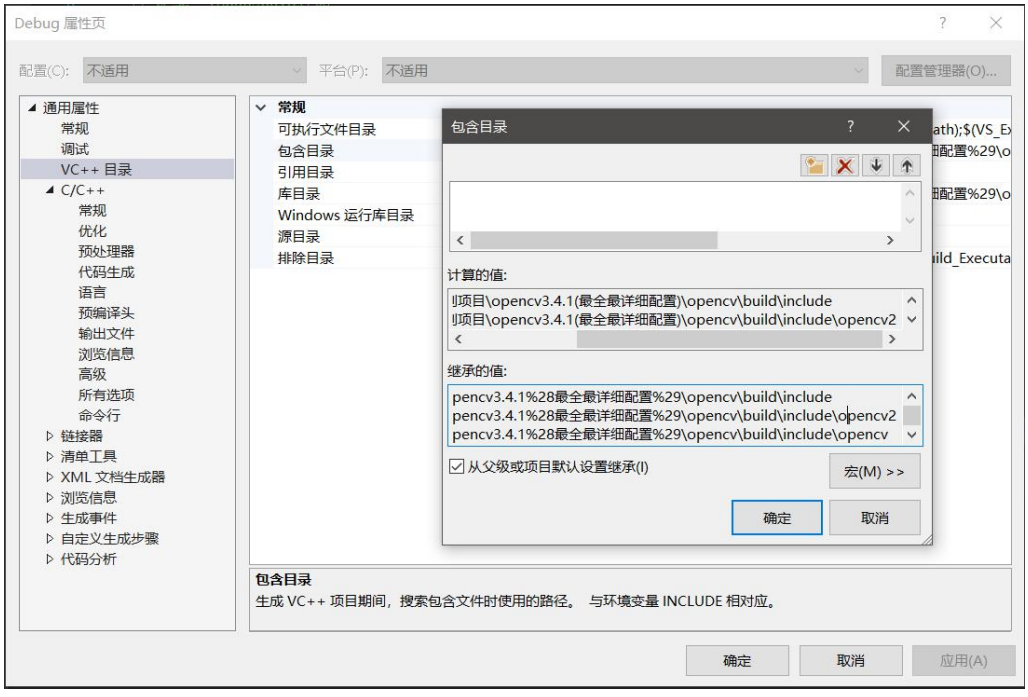

图 1.2 包含目录配置

3. 在选择库目录，向其中添加路径: opencv3.4.1(最全最详细配置)\opencv\build\x64\vc14\lib。如图 1.3 所示。

4. 再点击链接器-->输入，添加附加依赖项，输入 opencv_world341d.lib，opencv_world341.lib。如图 1.4 所示，最后单击确认。至此 opencv 环境配置完成。你可以通过引用 opencv 头文件进行测试是否配置成功(注意:编译与运行时将解决方案平台改为 x64)。如图 1.5 所示，#include <opencv2/opencv.hpp>编译后无报错即配置成功。

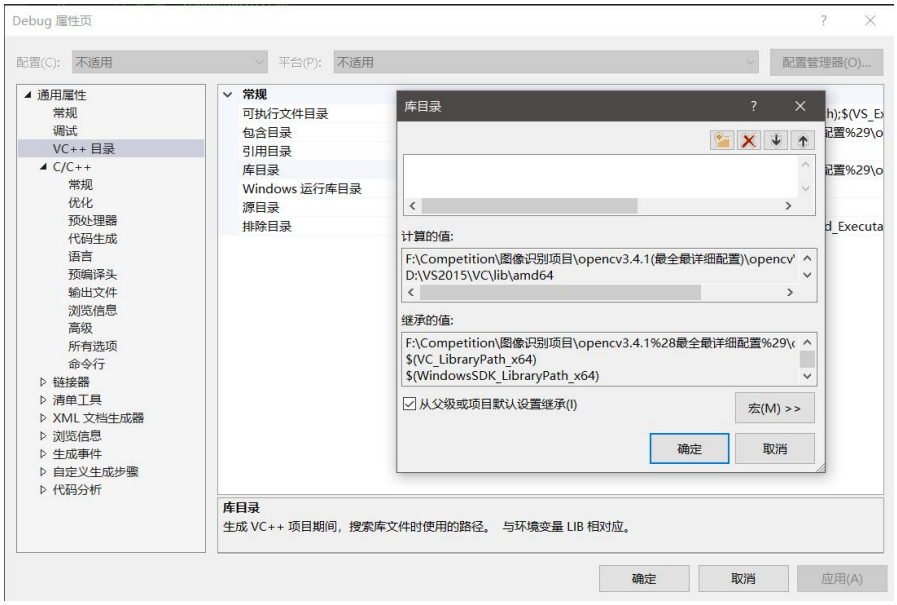

图 1.3 配置库目录

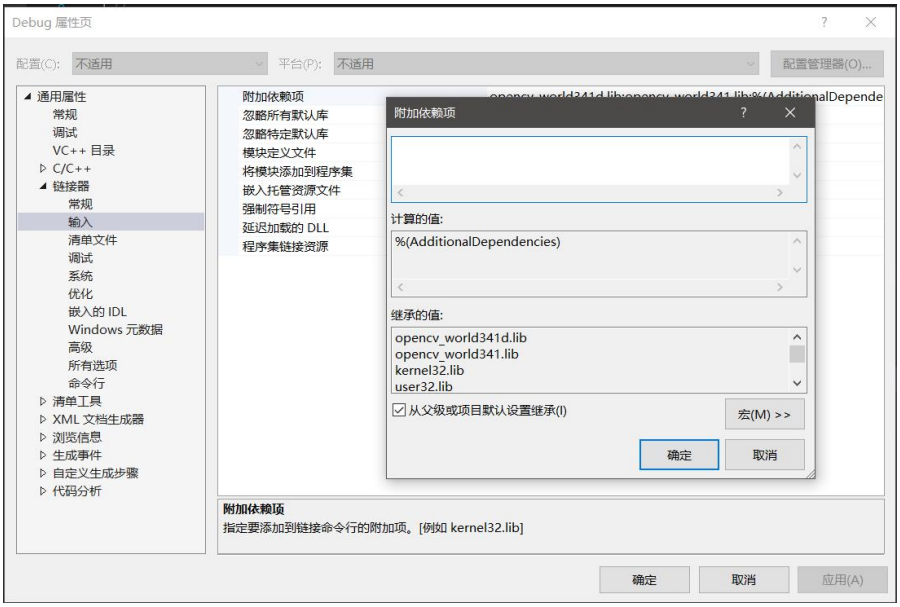

图 1.4 配置附加依赖项

图 1.5 正确引用头文件

5. 接下来配置读取示数的 API。将文件夹中的 Identify5VReadingsAPI.cpp 与 Identify5VReadingsAPI.h 文件复制到你的项目中，在解决方案管理器中右击项目名称。选择 C/C++编辑附加包括目录，将你所复制文件的路径添加其中，单击确定即可，如图 1.6。至此项目配置完成，你可以通过引用头文件： #include "Identify5VReadingsAPI.cpp" 。编译后不报错即配置成功，如图 1.7 所示。

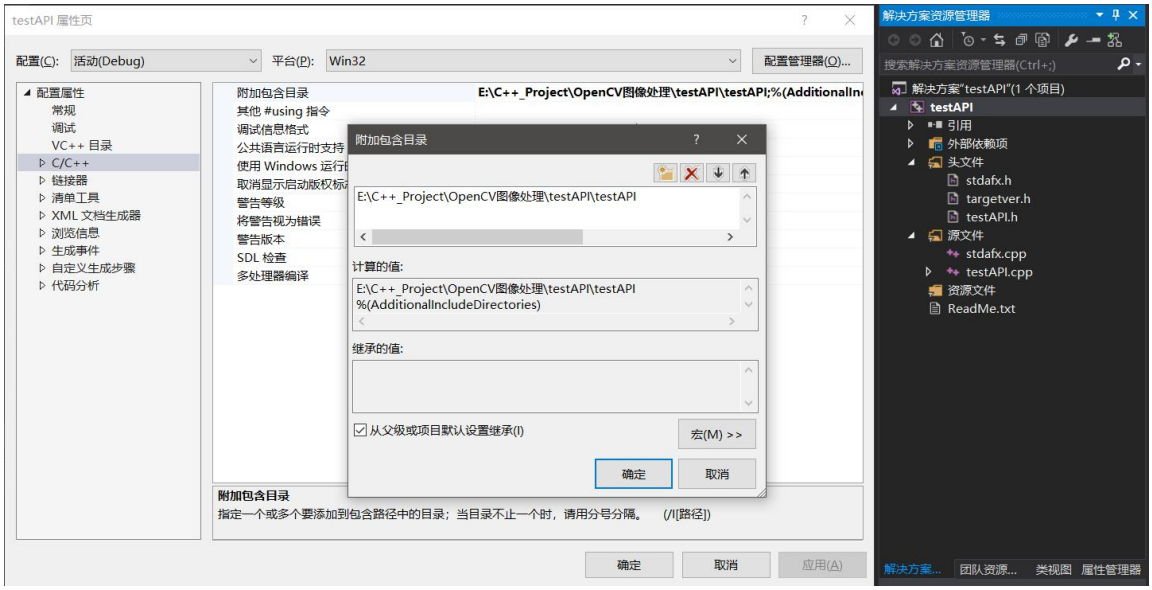

图 1.6 正确引用头文件

```
#include "Identify5VReadingsAPI.cpp"
```

图 1.7 正确引用头文件

二、API 函数

函数名	含义
int loadImg(string path , bool show)	加载图片，第一个参数为图片路径，第二个参数为是否以对话框的形式显示图片。
int loadImg(string path, bool show, const cv::String winname)	加载图片，前两个参数同上，第三个参数为显示对话框的名称。
int returnRows()	返回当前图像的行数。
int returnCols()	返回当前函数的列数。
int returnVid()	返回识别电压表的 ID，其返回值为 0,1,2,3。
double linearfittingVvalue()	利用最小二乘法实现直线拟合，最终获取的函数值返回为电压表示数。
double twopointVvalue()	利用两点法获得直线的斜率，最终获得电压表示数，相较于直线拟合法速度更快，但没有直线拟合法更准确。

三、使用方法

常用使用步骤：

1. 加载图片；
2. 查看是否能够识别到电压表，返回表 ID；
3. 识别到电压表后进行读数，返回示数。

简单示例：

```
#include "stdafx.h"
//识别示数的头文件
#include "Identify5VReadingsAPI.cpp"

int main()
{
    // 加载图片
    loadImg("E:/图片/OpenCV 图片/a2.bmp", 0);
    // 返回是否识别到电压表
    int i = returnVid();
    while (1)
    {
        cout <<"电压表为:表"<< i << endl;
        if (i <= 3 && i >= 1)
        {
            // 通过控制台不断输出电压表的值
            double i2 = twopointVvalue();
            cout << i2 << endl;
        }
        else
        {
            cout << "不能识别出电压表!" << endl;
            Sleep(1000);
        }
    }
    return 0;
```

四、其他说明

该 API 暂时还存在问题，还需要继续升级改造后才能准确的识别使用。