基于机器视觉的钢板尺寸测量软件

设计报告

目录

设计内容要求

设计一个软件系统，仿真钢板在线测量。基于多相机获取的钢板图像文件实现在钢板在线测量的仿真，具体内容包括：

（1）从读入的多个钢板图像文件，基于边缘检测识别钢板目标；

（2）基于识别结果，计算钢板的长度、宽度，实时显示在界面；

（3）实时显示钢板轮廓；

（4）实现单个相机从像素坐标到世界坐标的变换；

设计步骤和原理

分析要求，软件的设计分为以下几个功能的依次实现：

Button1

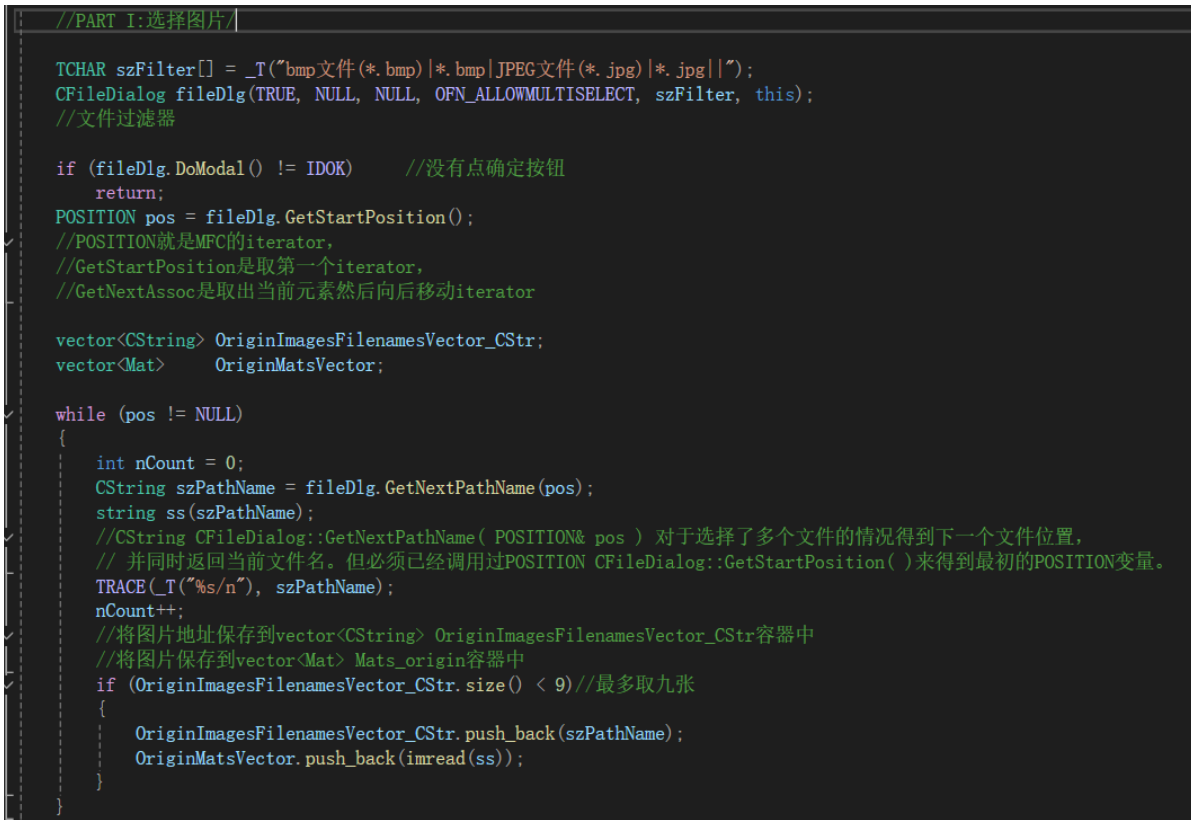
读入多个图像文件——识别钢板目标——处理含钢板的图片、用外接矩形逼近识别出的钢板轮廓——将钢板轮廓绘出来并显示——基于外接矩形与相机编号计算得出钢板的真正长宽——基于像素坐标在一个新图框里绘制出完整的钢板.

Button2

据输入的相机编号将输入的 x,y 坐标转化为世界坐标.

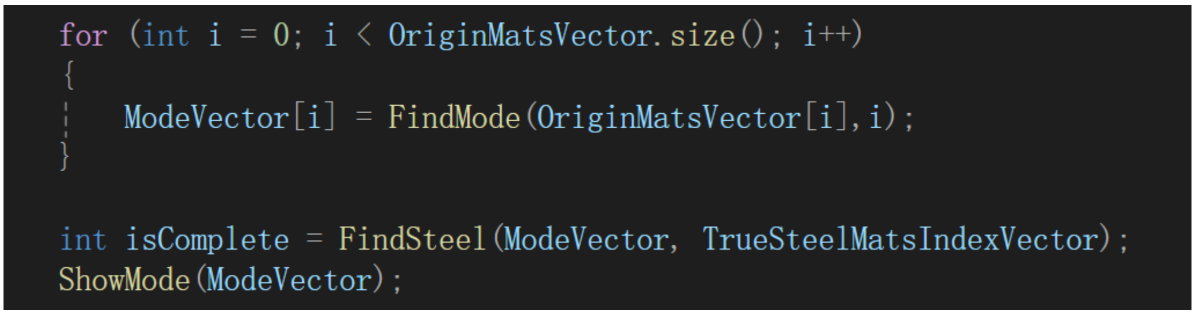
各个功能的简单原理和对应接口：

（1）读入多个图像文件——文件过滤器下打开一个文件对话框，最多将选择的九张图片的路径和 Mat 保存在两个 vector 中，待后续处理。



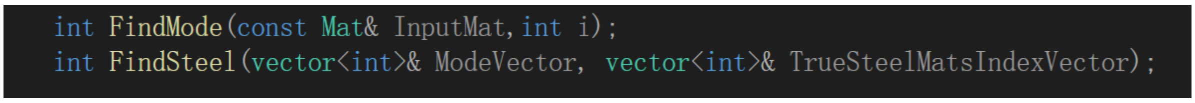
```
//PART I:选择图片/

TCHAR szFilter[] = _T("bmp文件(*.bmp)|*.bmp|JPEG文件(*.jpg)|*.jpg||");
CFileDialog fileDlg(TRUE, NULL, NULL, OFN_ALLOWMULTISELECT, szFilter, this);
//文件过滤器

if (fileDlg.DoModal() != IDOK)     //没有点确定按钮
    return;
POSITION pos = fileDlg.GetStartPosition();
//POSITION就是MFC的iterator,
//GetStartPosition是取第一个iterator,
//GetNextAssoc是取出当前元素然后向后移动iterator

vector<CString> OriginImagesFilenamesVector_CStr;
vector<Mat>     OriginMatsVector;

while (pos != NULL)
{
    int nCount = 0;
    CString szPathName = fileDlg.GetNextPathName(pos);
    string ss(szPathName);
    //CString CFileDialog::GetNextPathName( POSITION& pos ) 对于选择了多个文件的情况得到下一个文件位置,
    // 并同时返回当前文件名。但必须已经调用过POSITION CFileDialog::GetStartPosition( )来得到最初的POSITION变量。
    TRACE(_T("%s/n"), szPathName);
    nCount++;
    //将图片地址保存到vector<CString> OriginImagesFilenamesVector_CStr容器中
    //将图片保存到vector<Mat> Mats_origin容器中
    if (OriginImagesFilenamesVector_CStr.size() < 9)//最多取九张
    {
        OriginImagesFilenamesVector_CStr.push_back(szPathName);
        OriginMatsVector.push_back(imread(ss));
    }
}
```

（2）识别钢板目标——把每张图片划分左、中、右三个区域，转化为二值图，根据相机编号和三个区域灰度平均值比较从而识别出图片的四种模式（FindMode 实现）：钢板的头（mode=1）、中(mode=2)、尾(mode=3)、空(mode=0)。将序列图片模式按顺序存于一个 vector 数组中，并显示在界面上（ShowMode 实现）。对模式数组进行处理，用另一个 vector 存钢板头所在的索引与钢板尾所在的索引（FindSteel 实现），如果合理，说明识别出钢板目标。具体代码：



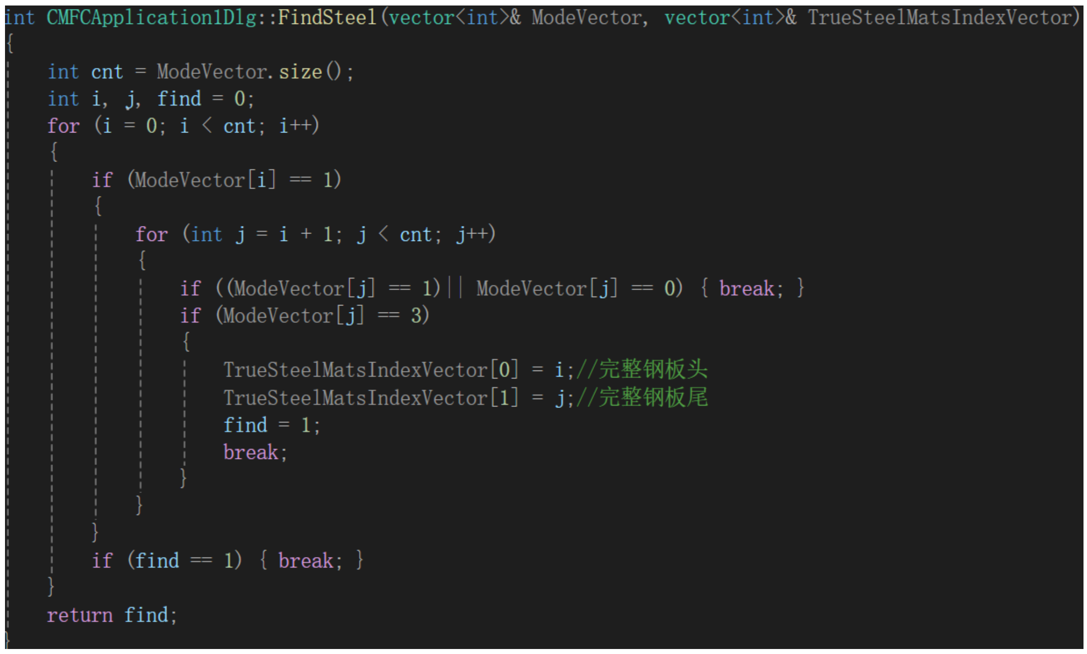
```
for (int i = 0; i < OriginMatsVector.size(); i++)
{
    ModeVector[i] = FindMode(OriginMatsVector[i],i);
}

int isComplete = FindSteel(ModeVector, TrueSteelMatsIndexVector);
ShowMode(ModeVector);
```

所用的函数签名



```
int FindMode(const Mat& InputMat,int i);
int FindSteel(vector<int>& ModeVector, vector<int>& TrueSteelMatsIndexVector);
```

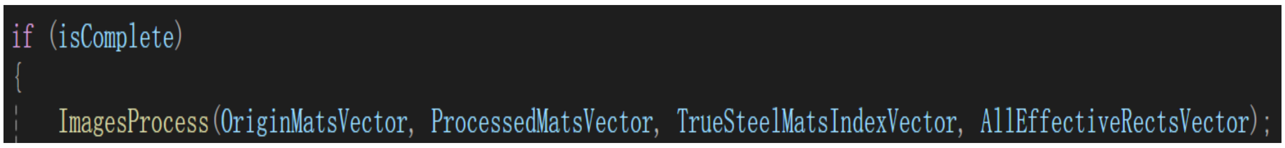
FindSteel 的定义



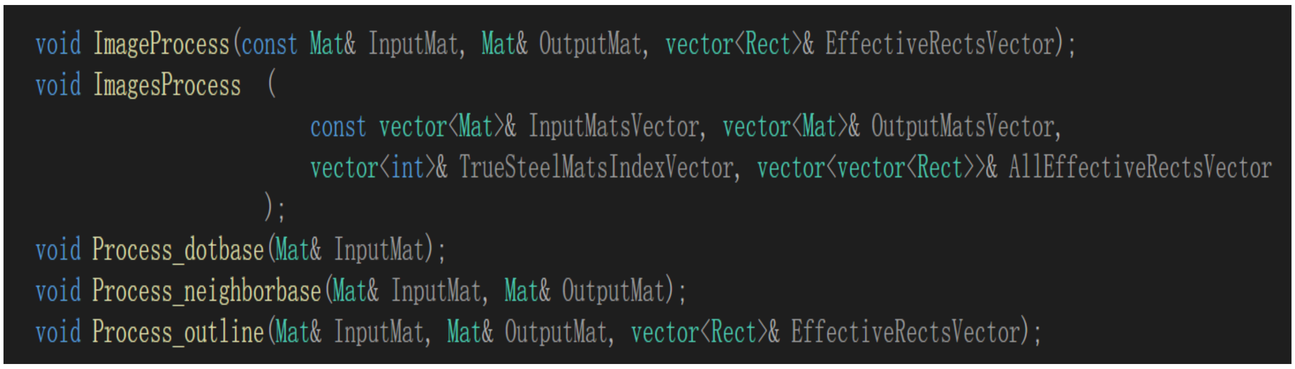
```
int CMFCApplication1Dlg::FindSteel(vector<int>& ModeVector, vector<int>& TrueSteelMatsIndexVector)
{
    int cnt = ModeVector.size();
    int i, j, find = 0;
    for (i = 0; i < cnt; i++)
    {
        if (ModeVector[i] == 1)
        {
            for (int j = i + 1; j < cnt; j++)
            {
                if ((ModeVector[j] == 1)|| ModeVector[j] == 0) { break; }
                if (ModeVector[j] == 3)
                {
                    TrueSteelMatsIndexVector[0] = i;//完整钢板头
                    TrueSteelMatsIndexVector[1] = j;//完整钢板尾
                    find = 1;
                    break;
                }
            }
        }
        if (find == 1) { break; }
    }
    return find;
```

FindMode 涉及到较复杂的数学运算，ShowMode 是简单的文本显示，具体定义请参阅附件代码。

（3）处理含钢板的图片、用外接矩形逼近识别出的钢板轮廓——首先进行基于点处理的亮度转换（Process_dotbase）、基于邻域的滤波处理(Process_neighborbase)、二值化与边缘检测等，然后跟踪轮廓(Process_outline)，将合适的外接矩形存于一个二维 vector 中（有些 mode==2 的图片有两个小矩形）。具体代码：

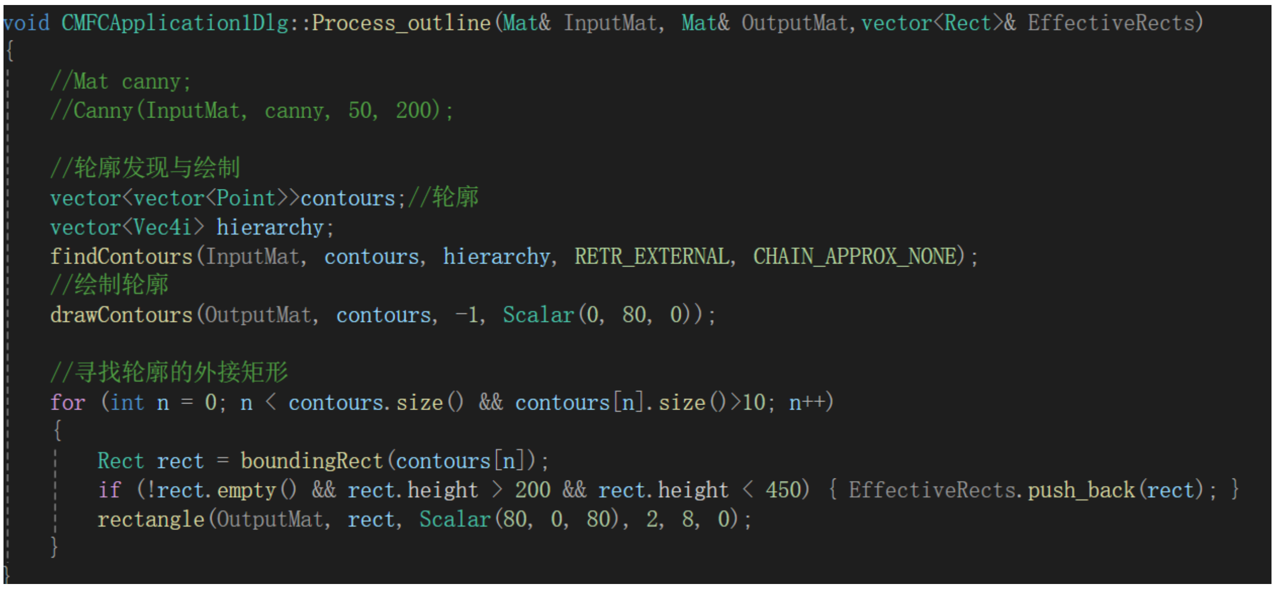

```
if (isComplete)
{
    ImagesProcess(OriginMatsVector, ProcessedMatsVector, TrueSteelMatsIndexVector, AllEffectiveRectsVector);
```

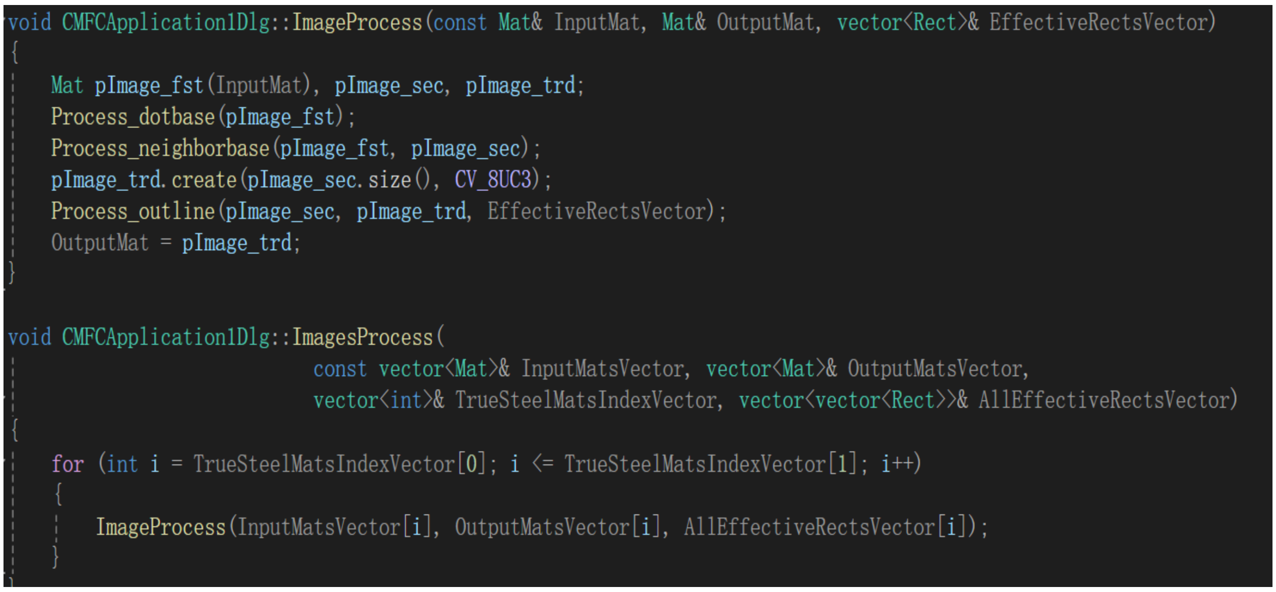
所用函数签名：


```
  void ImageProcess(const Mat& InputMat, Mat& OutputMat, vector<Rect>& EffectiveRectsVector);
  void ImagesProcess  (
                          const vector<Mat>& InputMatsVector, vector<Mat>& OutputMatsVector,
                          vector<int>& TrueSteelMatsIndexVector, vector<vector<Rect>>& AllEffectiveRectsVector
                      );
  void Process_dotbase(Mat& InputMat);
  void Process_neighborbase(Mat& InputMat, Mat& OutputMat);
  void Process_outline(Mat& InputMat, Mat& OutputMat, vector<Rect>& EffectiveRectsVector);
```

ImageProcess 调用所有 Process_...函数处理单张图片，ImagesProcess 对头索引至尾索引的图片进行批量处理。

部分函数定义：

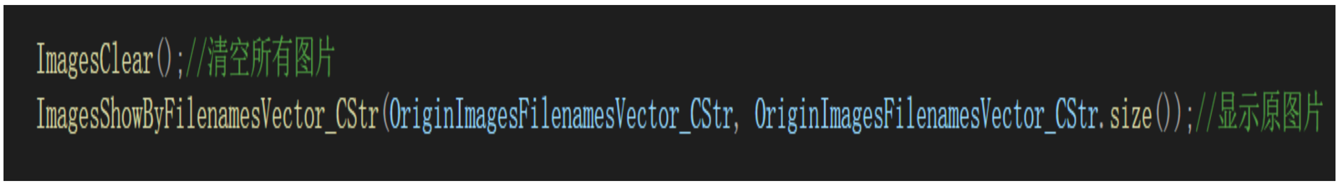

```
void CMFCApplication1Dlg::Process_outline(Mat& InputMat, Mat& OutputMat,vector<Rect>& EffectiveRects)
{
    //Mat canny;
    //Canny(InputMat, canny, 50, 200);

    //轮廓发现与绘制
    vector<vector<Point>>contours;//轮廓
    vector<Vec4i> hierarchy;
    findContours(InputMat, contours, hierarchy, RETR_EXTERNAL, CHAIN_APPROX_NONE);
    //绘制轮廓
    drawContours(OutputMat, contours, -1, Scalar(0, 80, 0));

    //寻找轮廓的外接矩形
    for (int n = 0; n < contours.size() && contours[n].size()>10; n++)
    {
        Rect rect = boundingRect(contours[n]);
        if (!rect.empty() && rect.height > 200 && rect.height < 450) { EffectiveRects.push_back(rect); }
        rectangle(OutputMat, rect, Scalar(80, 0, 80), 2, 8, 0);
    }
```



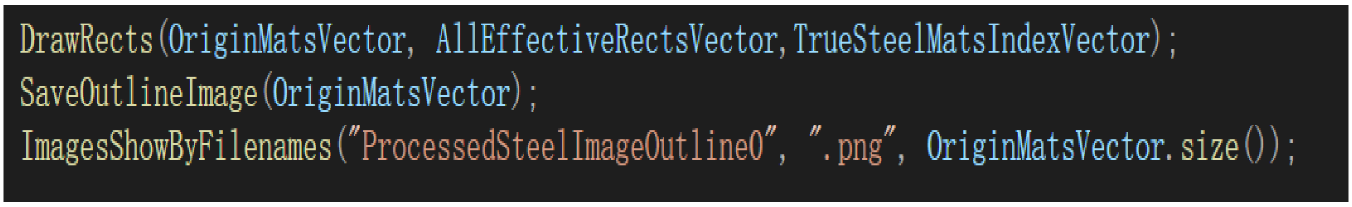
```
void CMFCApplication1Dlg::ImageProcess(const Mat& InputMat, Mat& OutputMat, vector<Rect>& EffectiveRectsVector)
{
    Mat pImage_fst(InputMat), pImage_sec, pImage_trd;
    Process_dotbase(pImage_fst);
    Process_neighborbase(pImage_fst, pImage_sec);
    pImage_trd.create(pImage_sec.size(), CV_8UC3);
    Process_outline(pImage_sec, pImage_trd, EffectiveRectsVector);
    OutputMat = pImage_trd;
}

void CMFCApplication1Dlg::ImagesProcess(
                            const vector<Mat>& InputMatsVector, vector<Mat>& OutputMatsVector,
                            vector<int>& TrueSteelMatsIndexVector, vector<vector<Rect>>& AllEffectiveRectsVector)
{
    for (int i = TrueSteelMatsIndexVector[0]; i <= TrueSteelMatsIndexVector[1]; i++)
    {
        ImageProcess(InputMatsVector[i], OutputMatsVector[i], AllEffectiveRectsVector[i]);
    }
}
```

亮度转换的 Process_dotbase 与滤波的 Process_neiborbase 只是简单地调用 opencv 库函数，具体定义请参与附件代码。

（4）将钢板轮廓绘出来并显示——基于已经保存的原始图像 Mat，钢板头尾索引和外接矩形，将外接矩形绘制到对应原始图像上去（DrawRects 实现），然后保存图像（SaveOutlineImage 实现），再通过读取对应路径文件将这些图像显示出来。在程序的开头清空图片控件（ImageClear 实现），然后显示原始图片(ImagesShowByFilenames_CStr 实现),有完整钢板则用处理后图片覆盖原图片（ImagesShowByFilenames 实现），否则不做操作。具体代码：

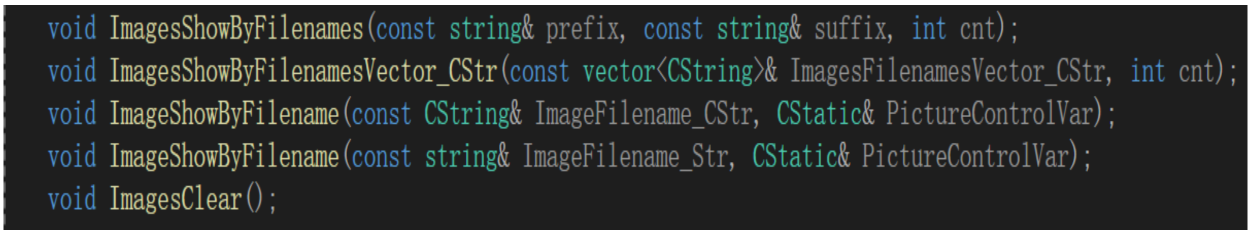


```
ImagesClear();//清空所有图片
ImagesShowByFilenamesVector_CStr(OriginImagesFilenamesVector_CStr, OriginImagesFilenamesVector_CStr.size());//显示原图片
```

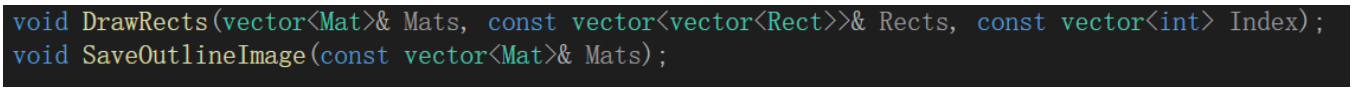


```
DrawRects(OriginMatsVector, AllEffectiveRectsVector,TrueSteelMatsIndexVector);
SaveOutlineImage(OriginMatsVector);
ImagesShowByFilenames("ProcessedSteelImageOutline0", ".png", OriginMatsVector.size());
```

所用的函数签名：

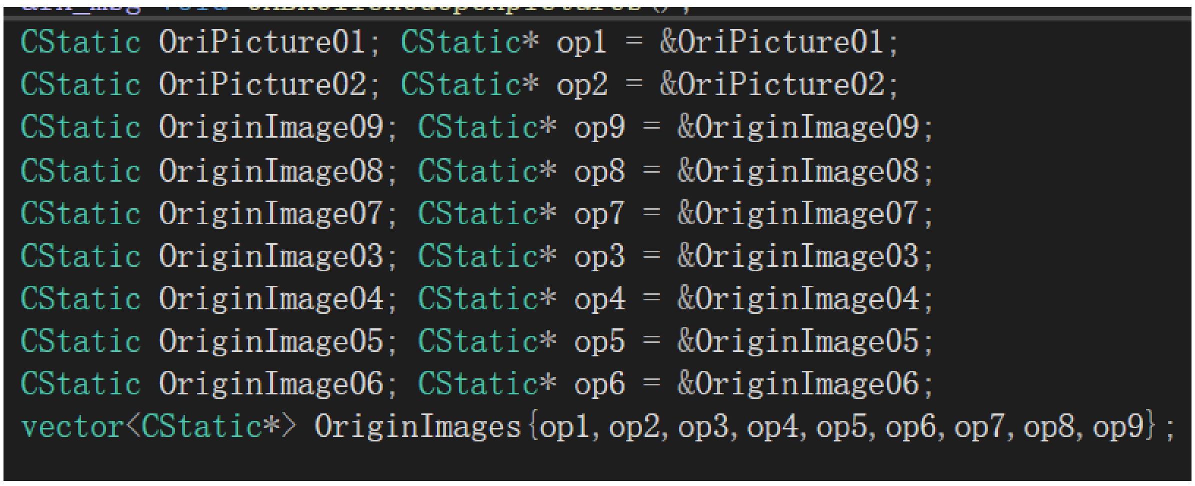


```
void ImagesShowByFilenames(const string& prefix, const string& suffix, int cnt);
void ImagesShowByFilenamesVector_CStr(const vector<CString>& ImagesFilenamesVector_CStr, int cnt);
void ImageShowByFilename(const CString& ImageFilename_CStr, CStatic& PictureControlVar);
void ImageShowByFilename(const string& ImageFilename_Str, CStatic& PictureControlVar);
void ImagesClear();
```

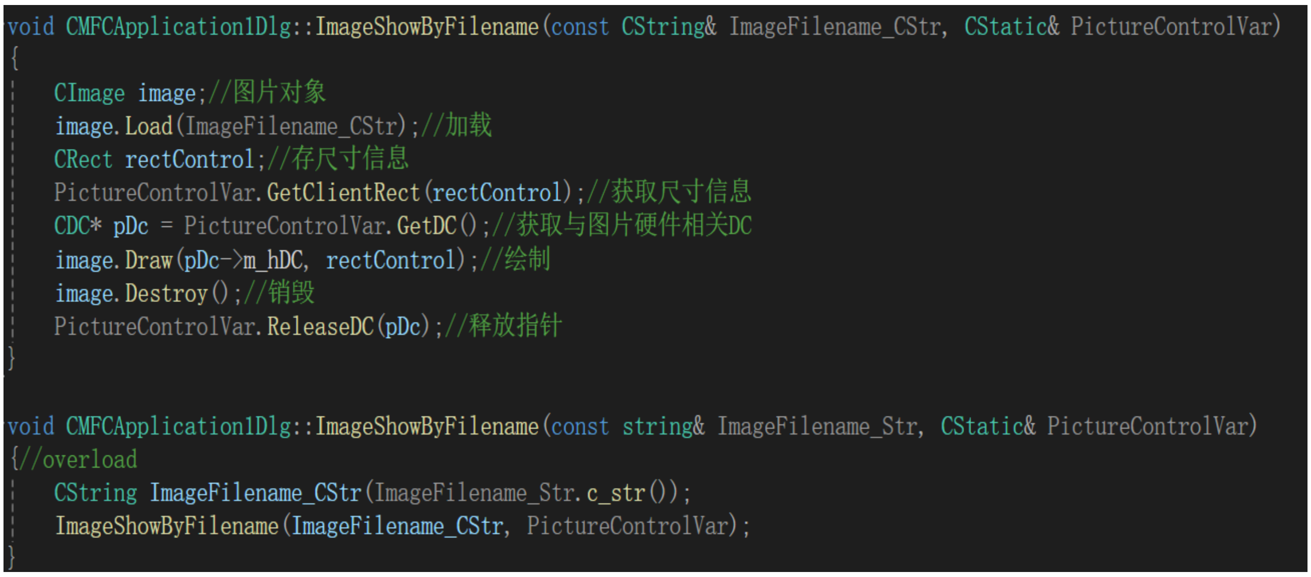


```
void DrawRects(vector<Mat>& Mats, const vector<vector<Rect>>& Rects, const vector<int> Index);
void SaveOutlineImage(const vector<Mat>& Mats);
```

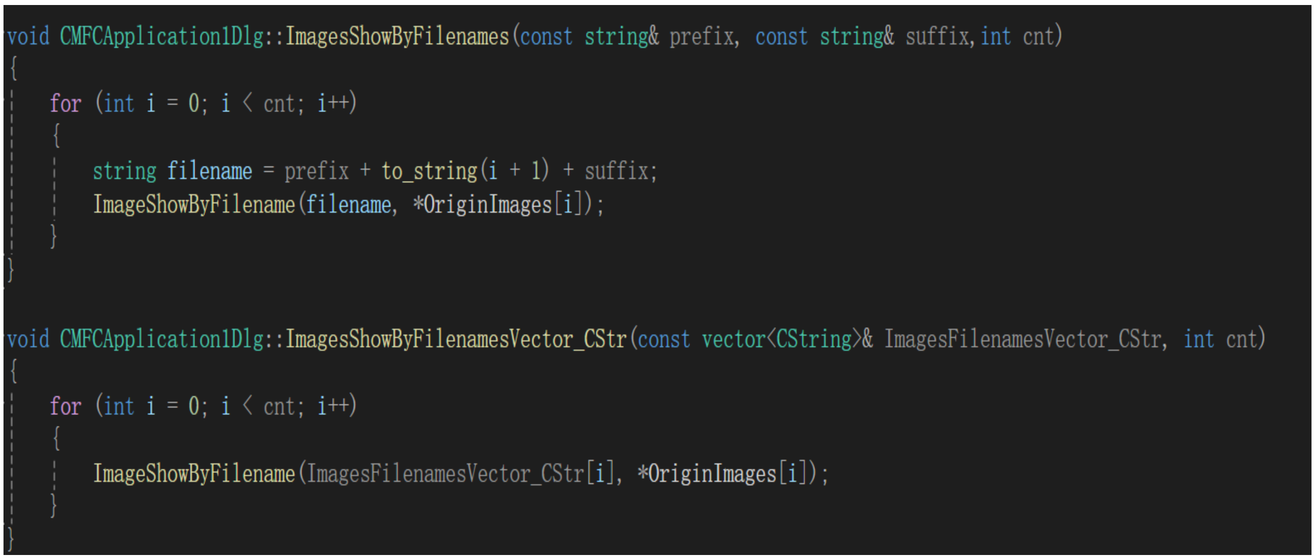
函数和部分变量定义：



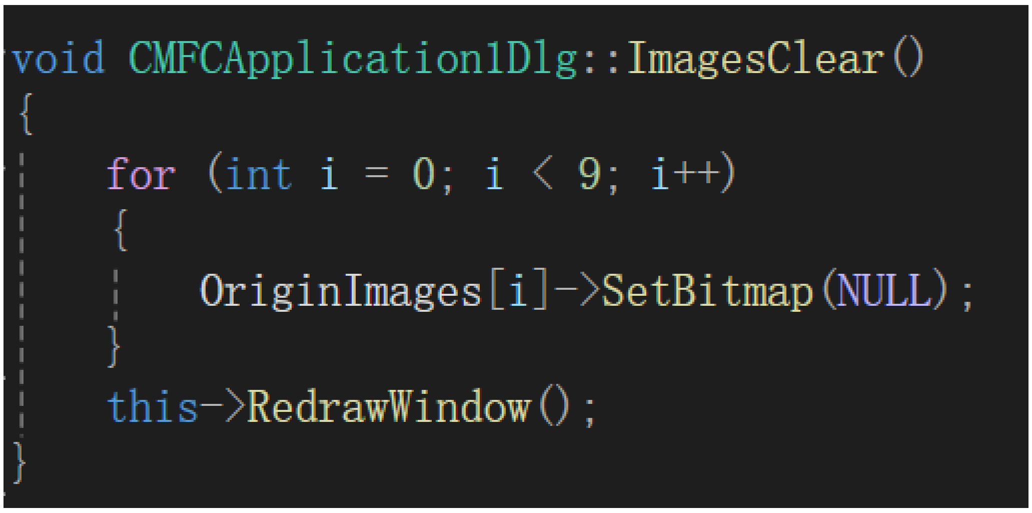
```
CStatic OriPicture01; CStatic* op1 = &OriPicture01;
CStatic OriPicture02; CStatic* op2 = &OriPicture02;
CStatic OriginImage09; CStatic* op9 = &OriginImage09;
CStatic OriginImage08; CStatic* op8 = &OriginImage08;
CStatic OriginImage07; CStatic* op7 = &OriginImage07;
CStatic OriginImage03; CStatic* op3 = &OriginImage03;
CStatic OriginImage04; CStatic* op4 = &OriginImage04;
CStatic OriginImage05; CStatic* op5 = &OriginImage05;
CStatic OriginImage06; CStatic* op6 = &OriginImage06;
vector<CStatic*> OriginImages{op1, op2, op3, op4, op5, op6, op7, op8, op9};
```



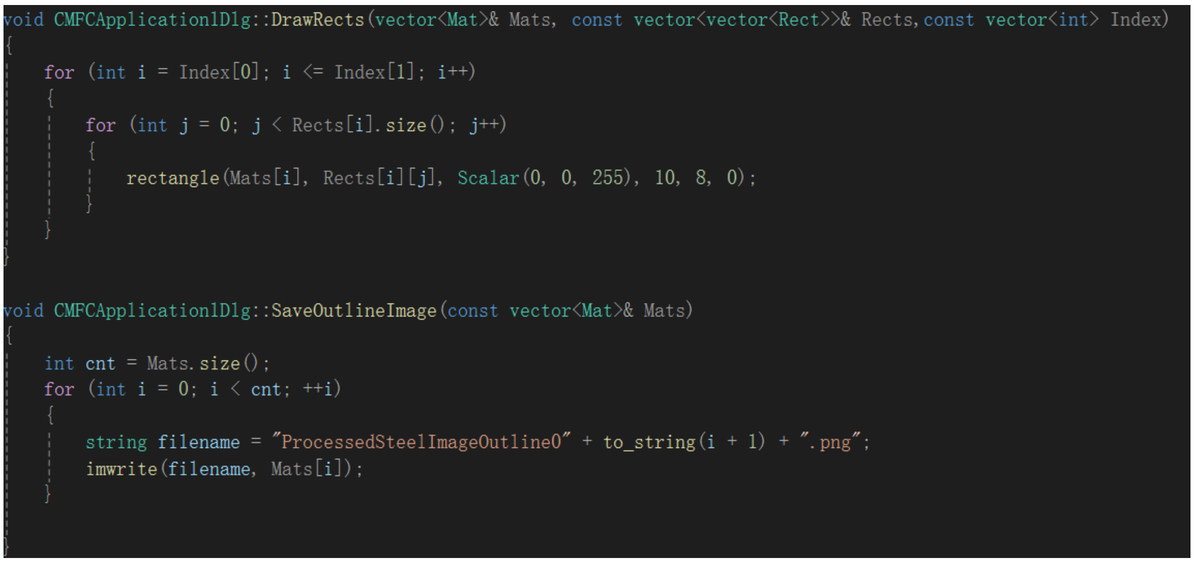
```
void CMFCApplication1Dlg::ImageShowByFilename(const CString& ImageFilename_CStr, CStatic& PictureControlVar)
{
    CImage image;//图片对象
    image.Load(ImageFilename_CStr);//加载
    CRect rectControl;//存尺寸信息
    PictureControlVar.GetClientRect(rectControl);//获取尺寸信息
    CDC* pDc = PictureControlVar.GetDC();//获取与图片硬件相关DC
    image.Draw(pDc->m_hDC, rectControl);//绘制
    image.Destroy();//销毁
    PictureControlVar.ReleaseDC(pDc);//释放指针
}

void CMFCApplication1Dlg::ImageShowByFilename(const string& ImageFilename_Str, CStatic& PictureControlVar)
{//overload
    CString ImageFilename_CStr(ImageFilename_Str.c_str());
    ImageShowByFilename(ImageFilename_CStr, PictureControlVar);
}
```



```
void CMFCApplication1Dlg::ImagesShowByFilenames(const string& prefix, const string& suffix,int cnt)
{
    for (int i = 0; i < cnt; i++)
    {
        string filename = prefix + to_string(i + 1) + suffix;
        ImageShowByFilename(filename, *OriginImages[i]);
    }
}

void CMFCApplication1Dlg::ImagesShowByFilenamesVector_CStr(const vector<CString>& ImagesFilenamesVector_CStr, int cnt)
{
    for (int i = 0; i < cnt; i++)
    {
        ImageShowByFilename(ImagesFilenamesVector_CStr[i], *OriginImages[i]);
    }
}
```

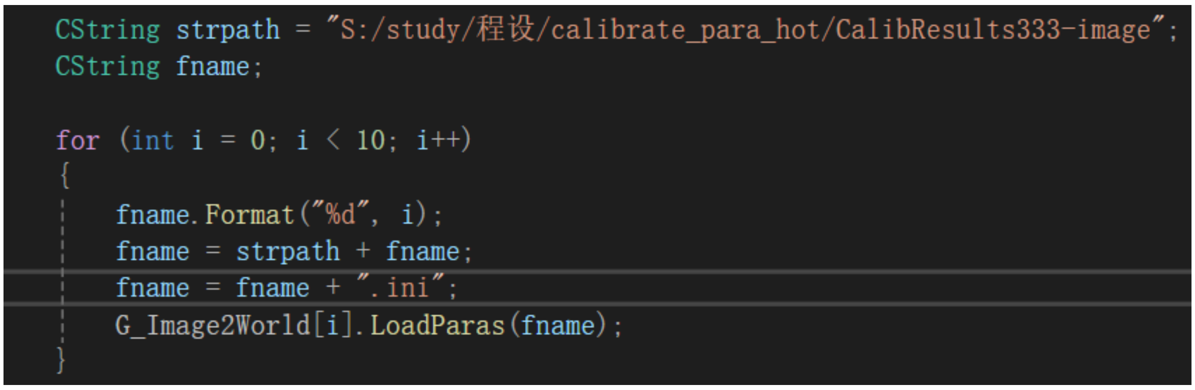


```
void CMFCApplication1Dlg::ImagesClear()
{
    for (int i = 0; i < 9; i++)
    {
        OriginImages[i]->SetBitmap(NULL);
    }
    this->RedrawWindow();
}
```

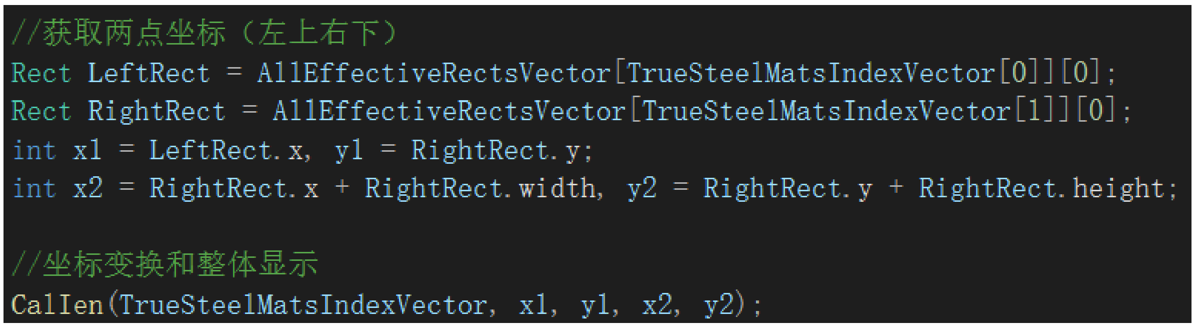


```
void CMFCApplication1Dlg::DrawRects(vector<Mat>& Mats, const vector<vector<Rect>>& Rects,const vector<int> Index)
{
    for (int i = Index[0]; i <= Index[1]; i++)
    {
        for (int j = 0; j < Rects[i].size(); j++)
        {
            rectangle(Mats[i], Rects[i][j], Scalar(0, 0, 255), 10, 8, 0);
        }
    }
}

void CMFCApplication1Dlg::SaveOutlineImage(const vector<Mat>& Mats)
{
    int cnt = Mats.size();
    for (int i = 0; i < cnt; ++i)
    {
        string filename = "ProcessedSteelImageOutline0" + to_string(i + 1) + ".png";
        imwrite(filename, Mats[i]);
    }

}
```

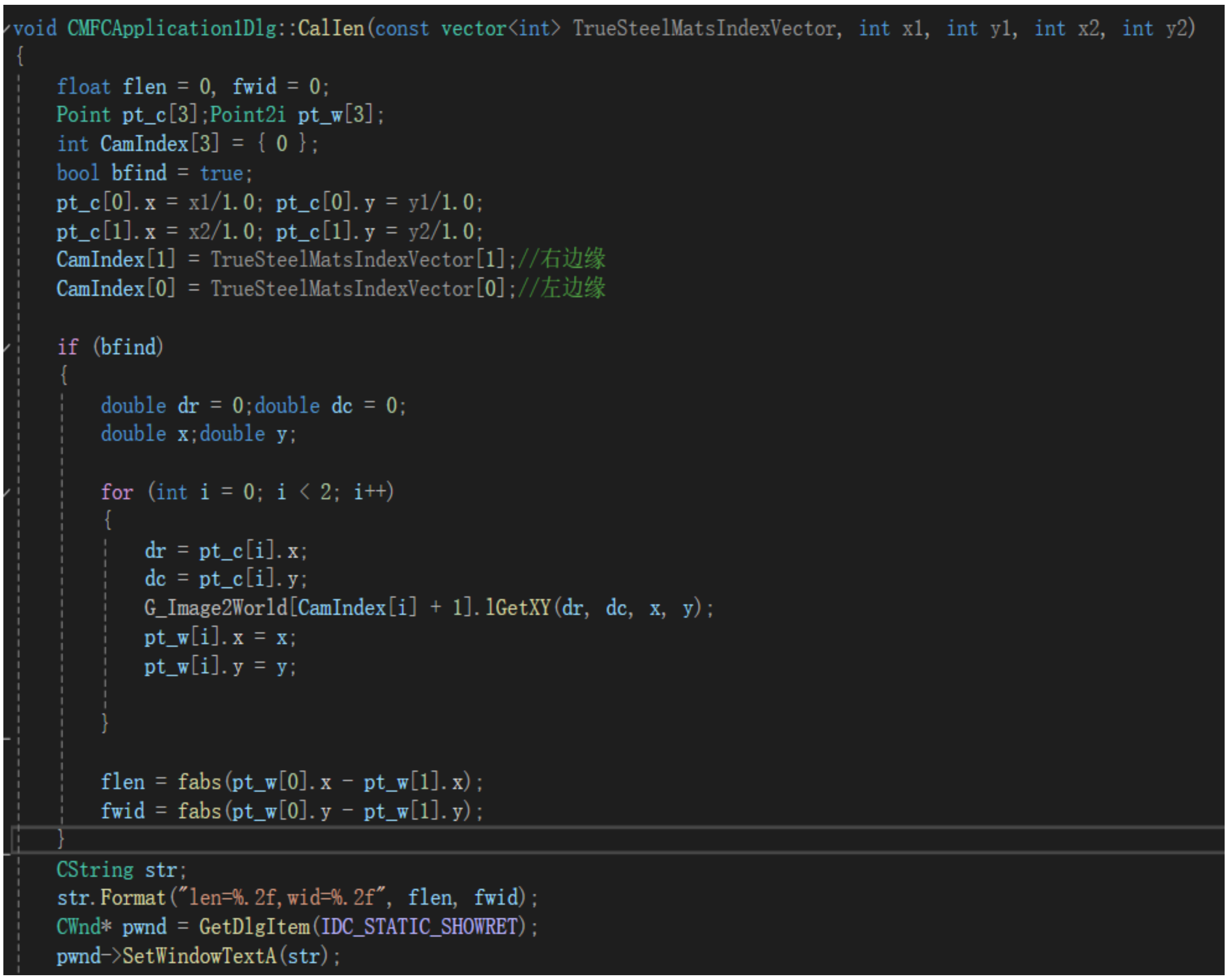
（5）基于外接矩形与相机编号计算得出钢板的真正长宽——首先引入标定文件，标定每个相机，标定结果存放在 G_Image2World 数组中，用 Rect 的成员变量得到钢板左上和右下两个坐标，与 G_Image2World 对接即可，具体代码：



```
CString strpath = "S:/study/程设/calibrate_para_hot/CalibResults333-image";
CString fname;

for (int i = 0; i < 10; i++)
{
    fname.Format("%d", i);
    fname = strpath + fname;
    fname = fname + ".ini";
    G_Image2World[i].LoadParas(fname);
}
```

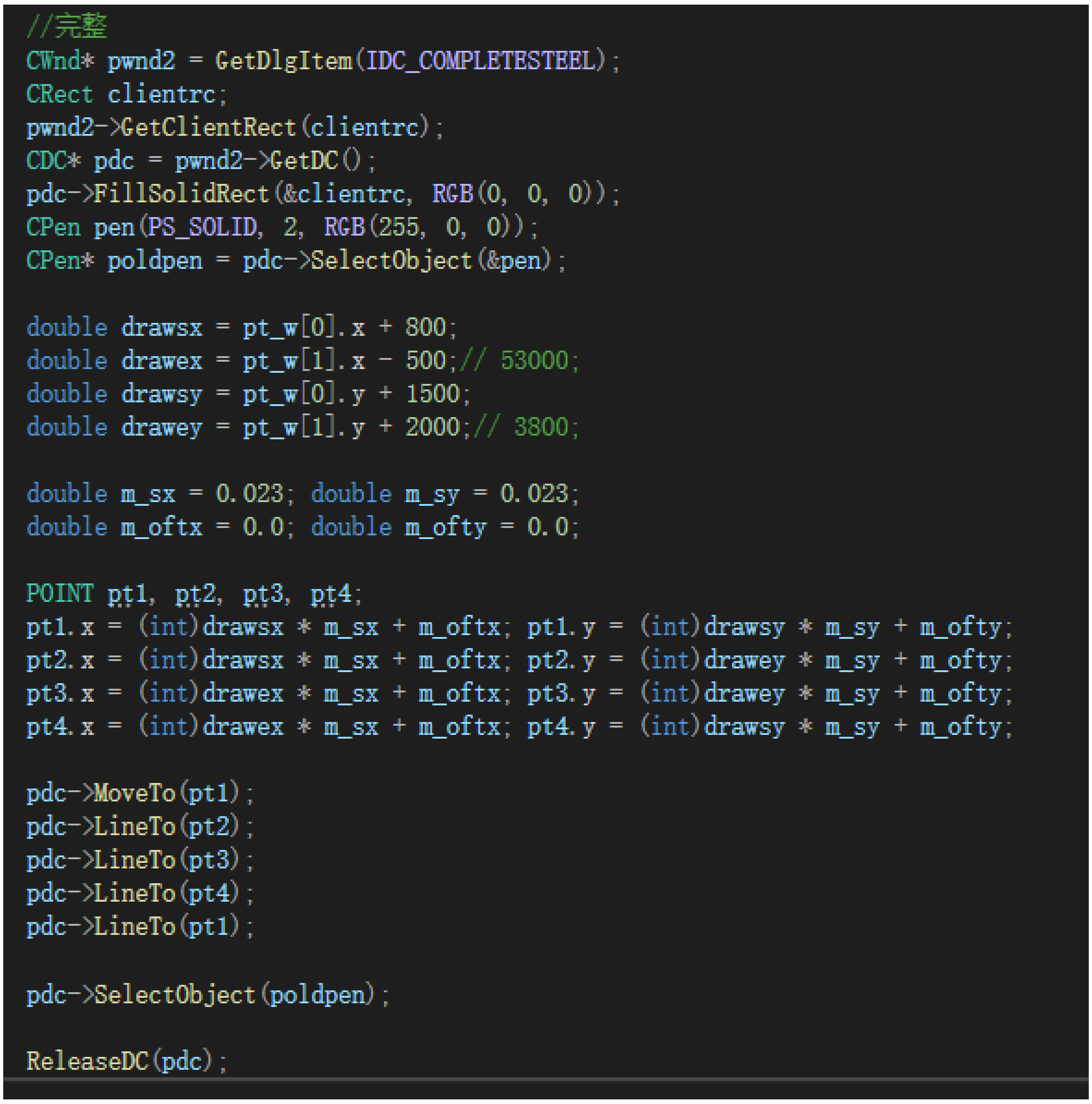


```
//获取两点坐标（左上右下）
Rect LeftRect = AllEffectiveRectsVector[TrueSteelMatsIndexVector[0]][0];
Rect RightRect = AllEffectiveRectsVector[TrueSteelMatsIndexVector[1]][0];
int x1 = LeftRect.x, y1 = RightRect.y;
int x2 = RightRect.x + RightRect.width, y2 = RightRect.y + RightRect.height;

//坐标变换和整体显示
CalIen(TrueSteelMatsIndexVector, x1, y1, x2, y2);
```



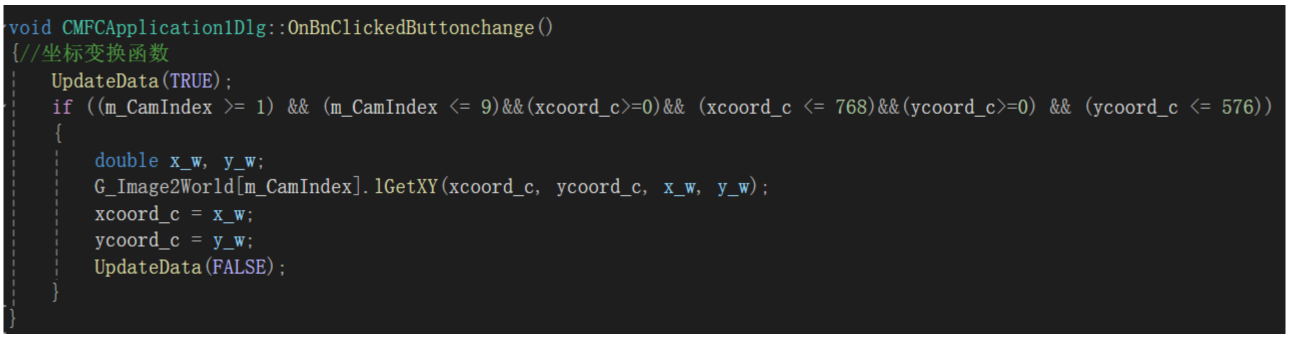
```
void CMFCApplication1Dlg::CalLen(const vector<int> TrueSteelMatsIndexVector, int x1, int y1, int x2, int y2)
{
    float flen = 0, fwid = 0;
    Point pt_c[3];Point2i pt_w[3];
    int CamIndex[3] = { 0 };
    bool bfind = true;
    pt_c[0].x = x1/1.0; pt_c[0].y = y1/1.0;
    pt_c[1].x = x2/1.0; pt_c[1].y = y2/1.0;
    CamIndex[1] = TrueSteelMatsIndexVector[1];//右边缘
    CamIndex[0] = TrueSteelMatsIndexVector[0];//左边缘

    if (bfind)
    {
        double dr = 0;double dc = 0;
        double x;double y;

        for (int i = 0; i < 2; i++)
        {
            dr = pt_c[i].x;
            dc = pt_c[i].y;
            G_Image2World[CamIndex[i] + 1].lGetXY(dr, dc, x, y);
            pt_w[i].x = x;
            pt_w[i].y = y;

        }

        flen = fabs(pt_w[0].x - pt_w[1].x);
        fwid = fabs(pt_w[0].y - pt_w[1].y);
    }
    CString str;
    str.Format("len=%.2f,wid=%.2f", flen, fwid);
    CWnd* pwnd = GetDlgItem(IDC_STATIC_SHOWRET);
    pwnd->SetWindowTextA(str);
```

（6）基于像素坐标在一个新图框里绘制出完整的钢板——在 CalIen 中已经得到了左上与右下的的世界坐标，对其进行平移与伸缩绘制在新图框即可。具体代码：



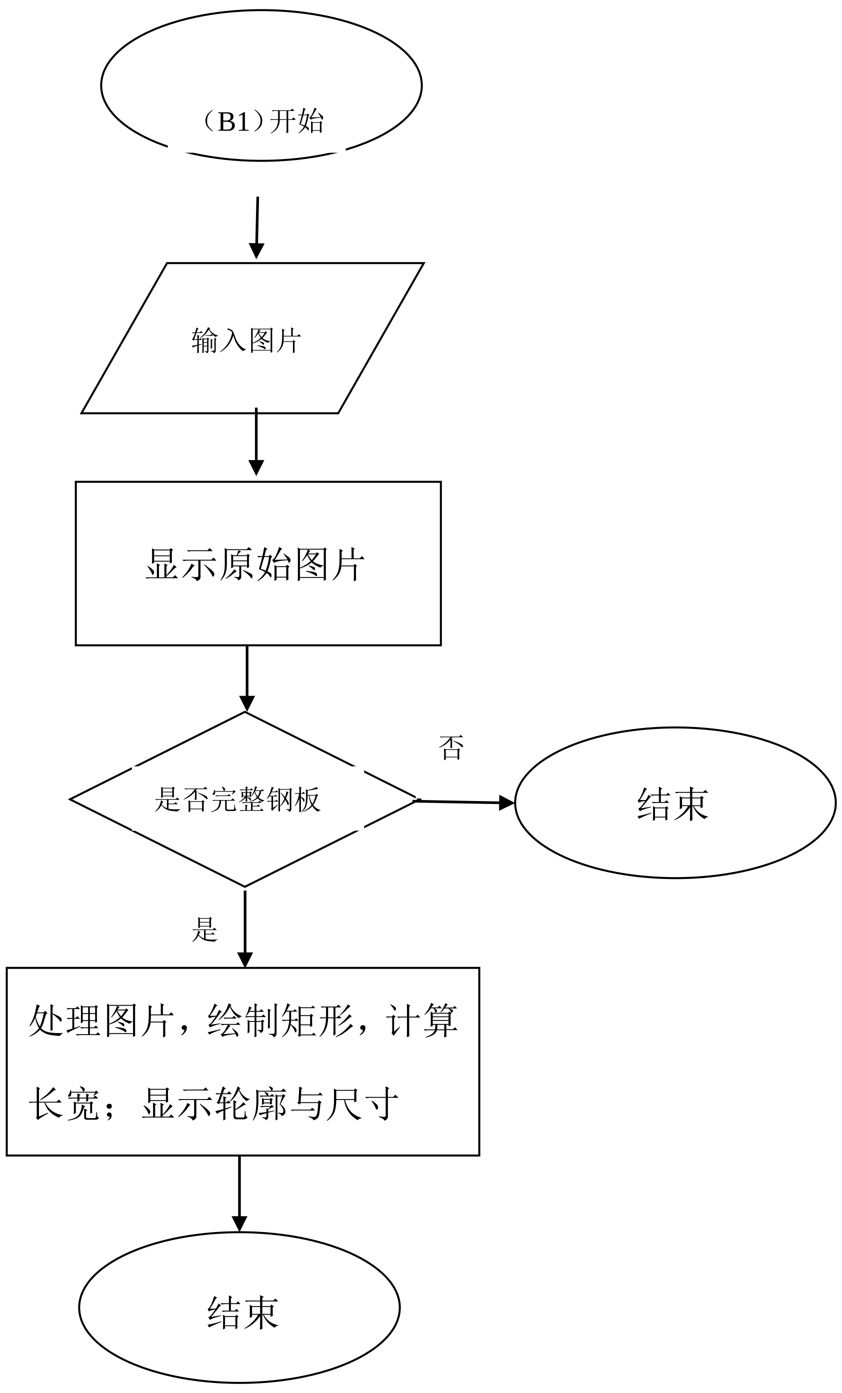
```
//完整
CWnd* pwnd2 = GetDlgItem(IDC_COMPLETESTEEL);
CRect clientrc;
pwnd2->GetClientRect(clientrc);
CDC* pdc = pwnd2->GetDC();
pdc->FillSolidRect(&clientrc, RGB(0, 0, 0));
CPen pen(PS_SOLID, 2, RGB(255, 0, 0));
CPen* poldpen = pdc->SelectObject(&pen);

double drawsx = pt_w[0].x + 800;
double drawex = pt_w[1].x - 500;// 53000;
double drawsy = pt_w[0].y + 1500;
double drawey = pt_w[1].y + 2000;// 3800;

double m_sx = 0.023; double m_sy = 0.023;
double m_oftx = 0.0; double m_ofty = 0.0;

POINT pt1, pt2, pt3, pt4;
pt1.x = (int)drawsx * m_sx + m_oftx; pt1.y = (int)drawsy * m_sy + m_ofty;
pt2.x = (int)drawsx * m_sx + m_oftx; pt2.y = (int)drawey * m_sy + m_ofty;
pt3.x = (int)drawex * m_sx + m_oftx; pt3.y = (int)drawey * m_sy + m_ofty;
pt4.x = (int)drawex * m_sx + m_oftx; pt4.y = (int)drawsy * m_sy + m_ofty;

pdc->MoveTo(pt1);
pdc->LineTo(pt2);
pdc->LineTo(pt3);
pdc->LineTo(pt4);
pdc->LineTo(pt1);

pdc->SelectObject(poldpen);

ReleaseDC(pdc);
```

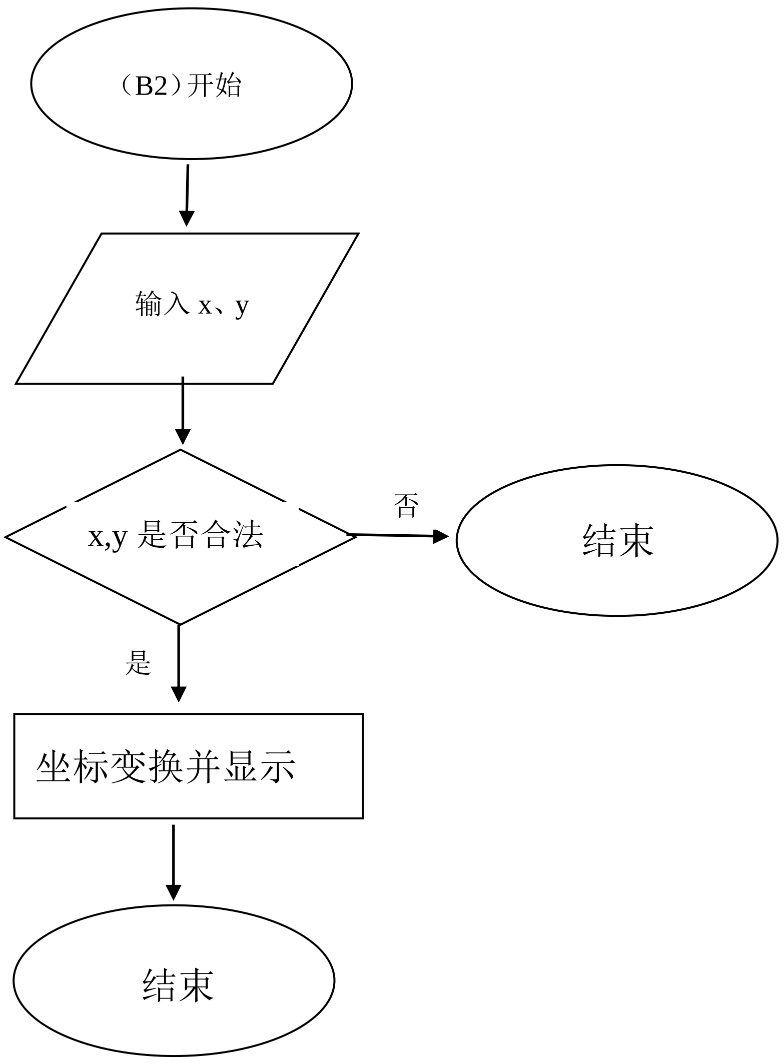
（7）据输入的相机编号将输入的 x,y 坐标转化为世界坐标——与（5）基本相同，加上 768*576 限制调用 Image2World 的函数即可。具体代码：



```
void CMFCApplication1Dlg::OnBnClickedButtonchange()
{//坐标变换函数
    UpdateData(TRUE);
    if ((m_CamIndex >= 1) && (m_CamIndex <= 9)&&(xcoord_c>=0)&& (xcoord_c <= 768)&&(ycoord_c>=0) && (ycoord_c <= 576))
    {
        double x_w, y_w;
        G_Image2World[m_CamIndex].lGetXY(xcoord_c, ycoord_c, x_w, y_w);
        xcoord_c = x_w;
        ycoord_c = y_w;
        UpdateData(FALSE);
    }
}
```

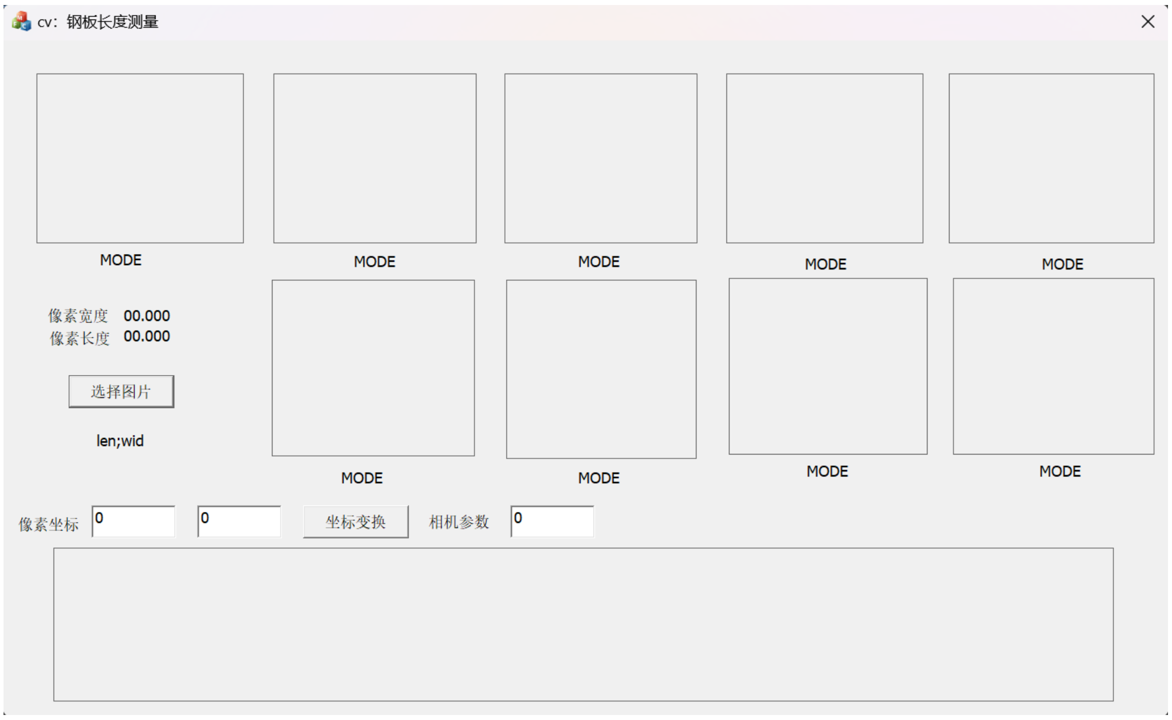
算法流程图



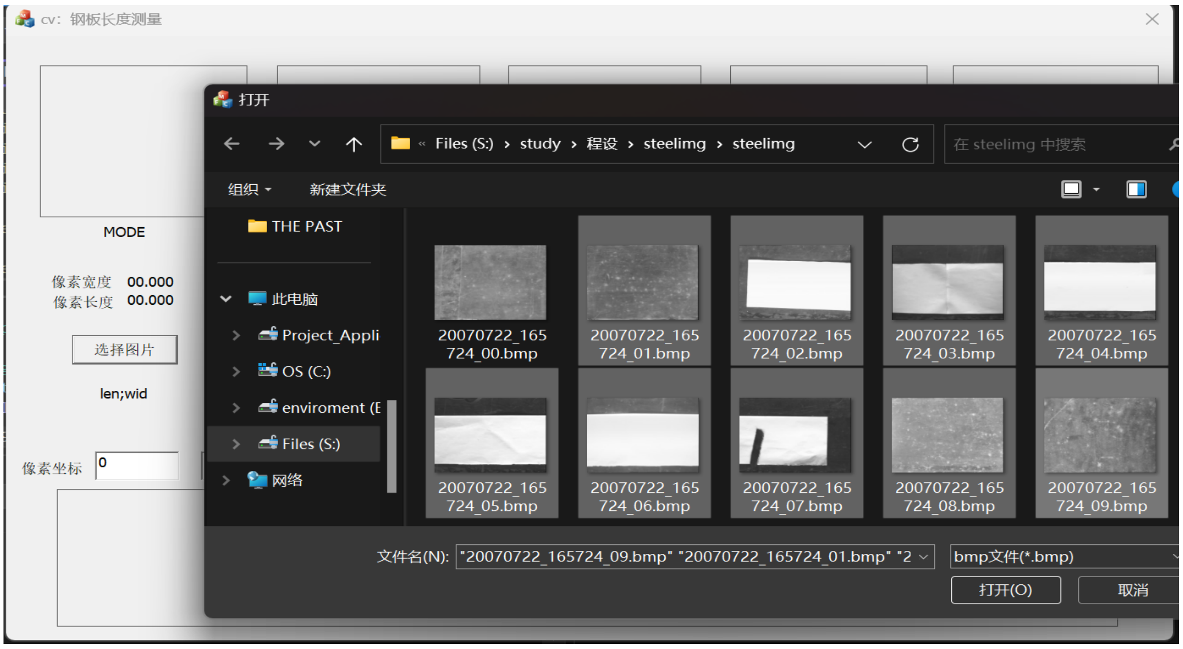

（B2）开始
输入 x、y
x,y 是否合法
否
结束
是
坐标变换并显示
结束

软件使用说明

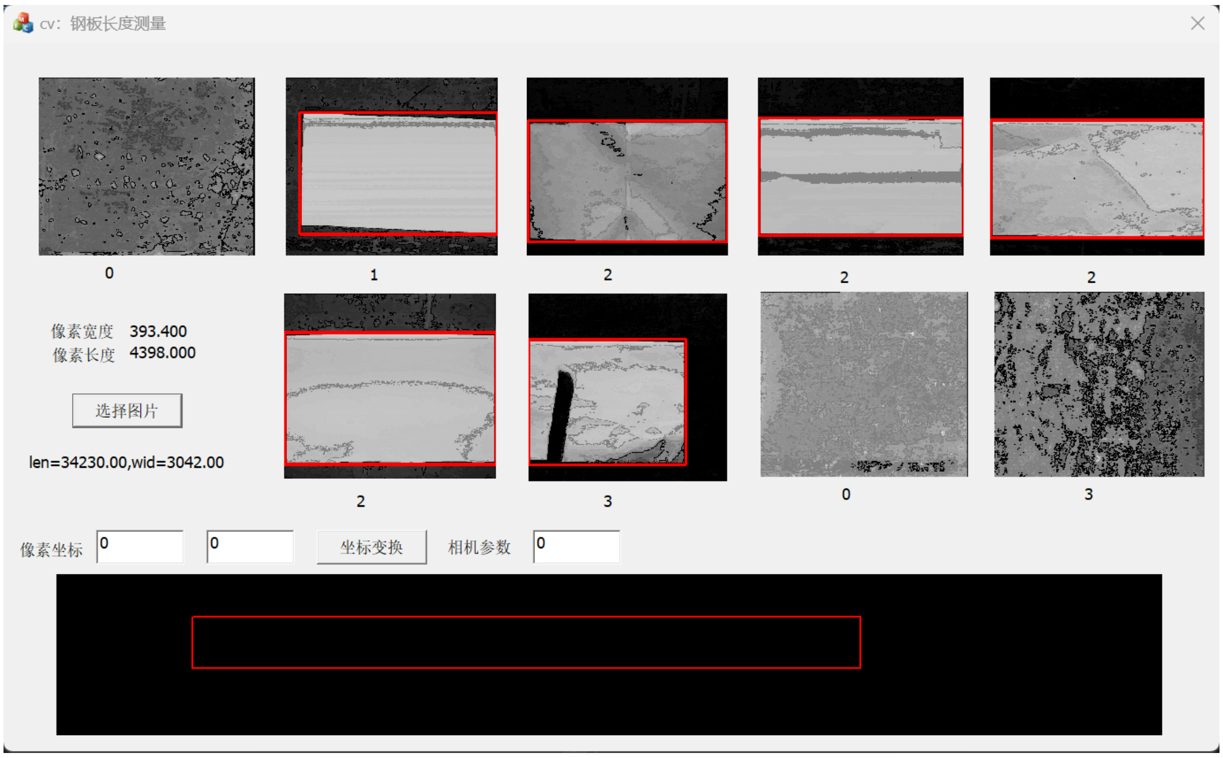
默认界面



单击“选择图片”，进入文件对话框，可选择 1~9 张图片



单击“打开”，可将选择的图片加载进软件，立得长宽与轮廓

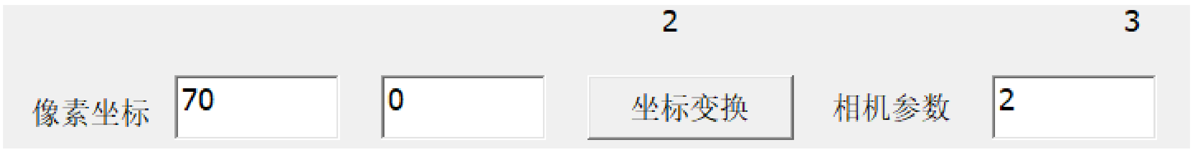

cv：钢板长度测量

像素宽度 393.400
像素长度 4398.000

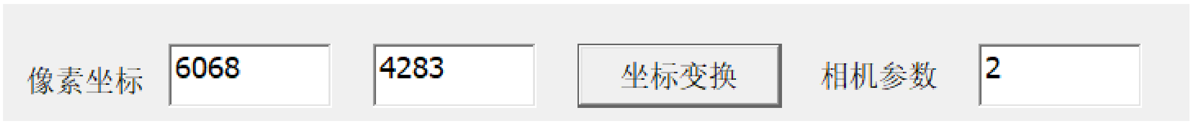
len=34230.00,wid=3042.00

像素坐标 0 0 坐标变换 相机参数 0

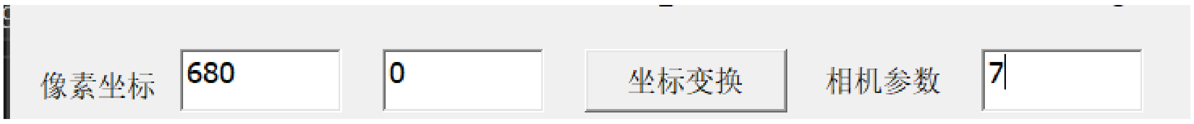
在像素坐标编辑框填入“70” “0”，相机参数“2”


像素坐标 70 0 坐标变换 相机参数 2

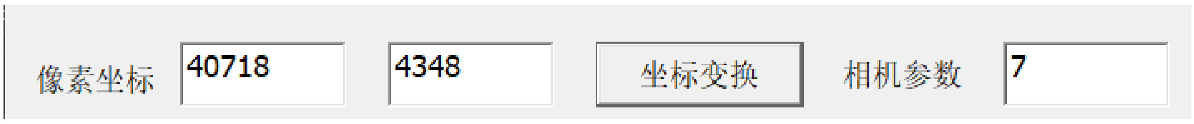
单击坐标变换，记录第一个编辑框数值


像素坐标 6068 4283 坐标变换 相机参数 2

在像素坐标编辑框填入“680” “0”，相机参数“7”


像素坐标 680 0 坐标变换 相机参数 7

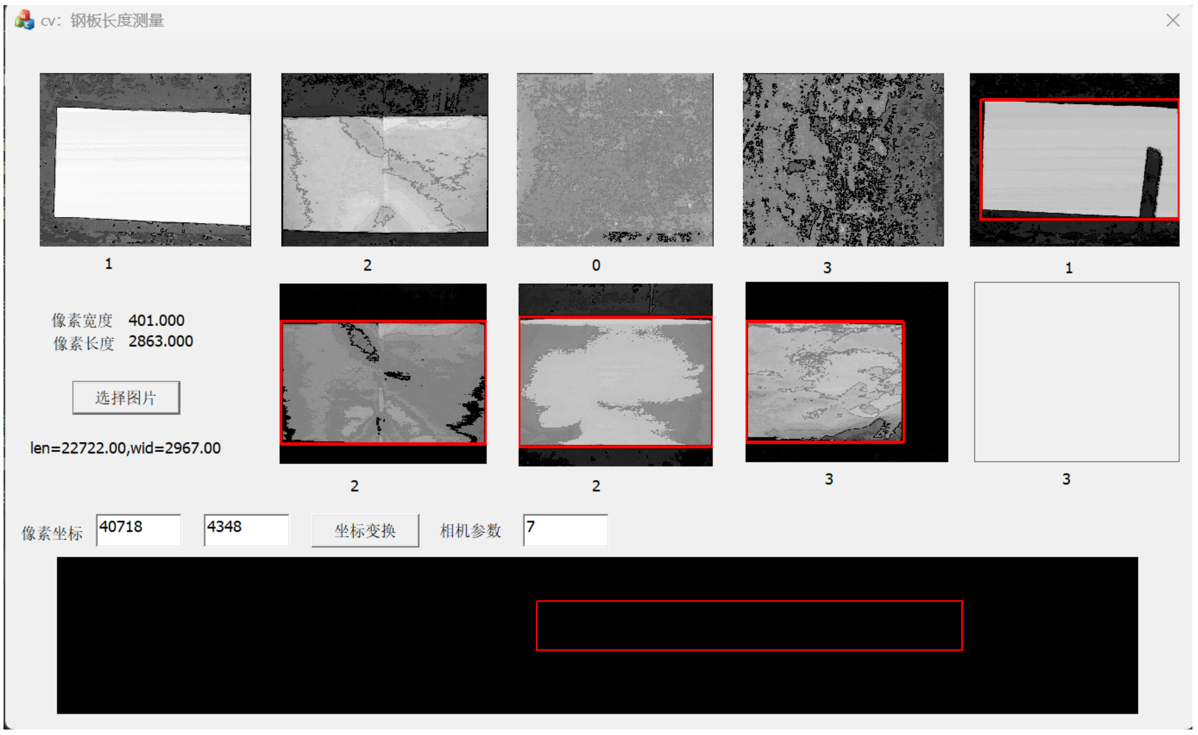
单击坐标变换，记录第一个编辑框数值


像素坐标 40718 4348 坐标变换 相机参数 7

40718-6068=34650，与测得的长度 34230 较接近，因此尺寸测量与坐标变

换是成功的。

本软件有强大的防干扰功能，如下

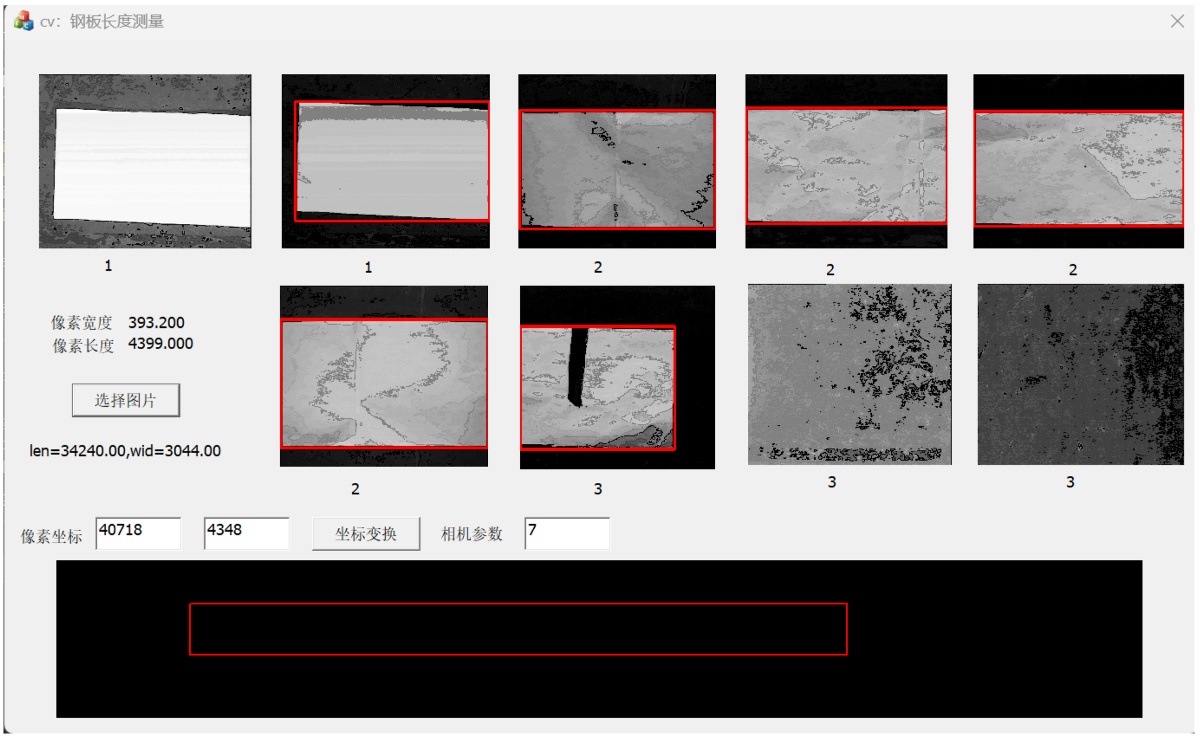


像素宽度 401.000
像素长度 2863.000

len=22722.00,wid=2967.00

像素坐标 40718 4348

相机参数 7



像素宽度 393.200
像素长度 4399.000

len=34240.00,wid=3044.00

像素坐标 40718 4348

相机参数 7

心得体会与课程建议

本次实训提高了我的程序设计能力。

（1）熟悉了 C++编程语言的基本语法，熟悉 STL（主要是 vector）的使用，熟悉 pointer 与 reference 的实际应用，熟悉了程序调试的基本步骤，对“强类型”有了更深刻的认识；

（2）在对 Mat、Rect 等 opencv 库中类的使用中加深了对面向对象程序设计（OOP）的体会，积累了模块/函数式编程的经验；

（3）学习了用 MFC 制作图形用户界面（GUI）并与后端对接的技术，同时激发了对时兴的 Qt 的兴趣；

（4）学习了用 OpenCV 库函数进行图像处理，如滤波 blur、二值化 threshold、边缘检测 Canny、轮廓提取 findContours 等等，初步接触计算机视觉；

（5）遇到了许多“调参”场景，如图像处理时的阈值、排除不适合的小矩形、缩放比例等等，这是不可避免的过程，但也没有完美的参数，这让我认识到可以为应用加一个自主调参的功能，以求面对不同图像时都能有最佳表现。

（6）掌握“调库”的操作，不仅可以随心所欲地用它解决问题，也可以通过阅读源码提升自己能力，是一个合格程序员必须掌握的能力。

参考文献

MFC：https://blog.csdn.net/qq_45549336/article/details/109008192

滤波算法：https://blog.csdn.net/obsorb_knowledge/article/details/100514115

轮廓提取：https://blog.csdn.net/great_yzl/article/details/119807705

Rect：https://blog.csdn.net/sinat_38102206/article/details/80996897