一、概述

使用面向对象语言 C++进行程序设计，编写出一个完整的特征点提取及相关系数法匹配程序对实验数据进行计算，输出特征点提取后图片以及影像匹配的窗口影像。实习目的为深入理解特征点提取与影像匹配的原理，通过自我编程加强动手能力的培养，通过对实验结果的分析，增强综合运用所学知识解决实际问题的能力。

原始数据：

u0367_panRight.tif/bmp

u0369_panLeft.tif/bmp

二、特征提取算法原理及算法

本次特征点提取使用了 Moravec 算法提取特征

Moravec 算法提取介绍：

Moravec 算子在四个方向上计算非归一化的影像局部灰度差方差，将最小值作为兴趣值测度。因此，该算子检测的特征点是那些影响强度值在每个方向上变化剧烈的点。

算法步骤：

1、 **首先计算各像素的兴趣值**。在以像素（r,c）为中心的 w×w 的影像窗口内，计算四个方向相邻像素灰度差的平方和。然后取四个方向上值最小者作为该像素的兴趣值。

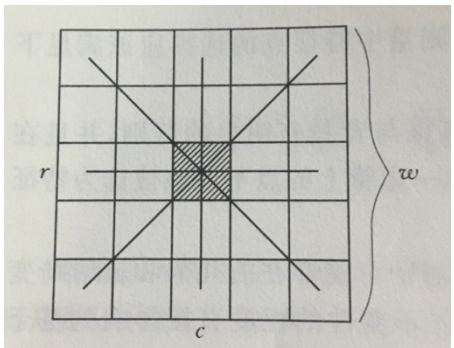


图 3-1-1 Moravec 算子计算示意图

$$V_1 = \sum_{i=-k}^{k-1} (g_{x+i,y} - g_{x+i+1,y})^2$$

$$V_2 = \sum_{i=-k}^{k-1} (g_{x+i,y+i} - g_{x+i+1,y+i+1})^2$$

$$V_3 = \sum_{i=-k}^{k-1} (g_{x,y+i} - g_{x,y+i+1})^2$$

$$V_4 = \sum_{i=-k}^{k-1} (g_{x+i,y-i} - g_{x+i+1,y-i-1})^2$$

2、 **给定阈值**，将兴趣值大于该阈值的点（即兴趣值计算窗口的中心点）作为候选点。阈值的选择以候选点中包括所需要的特征点而又不含过多的非特征点为原则。

3、 选候选点中兴趣值极大的点作为特征点。在一定大小的窗口内（可不同于兴趣值计算的窗口大小），将候选

点中兴趣值不是最大者去掉，仅留下兴趣值最大者，该像素即为特征点，该过程成为抑制局部非最大。

编程实现：（*重点函数）

遍历影像，寻找特征点：

```
void FeatureExtraction::GetFuature()
{

    int c = 0, r = 0;//初始化图像遍历位置

    FeaturePoint* temp = (FeaturePoint*)malloc(sizeof(FeaturePoint) * 1000000);

    for (c = int(w / 2); c < Img.rows - int(w / 2); c++) {
        for (r = int(w / 2); r < Img.cols - int(w / 2); r++) {
            double F = InterestValue(c, r);
            if (F >this->c_threshol) {
                //temp[numFE].value = InterestValue(c, r);
                temp[numFE].value = F;

                temp[numFE].rows = c;    temp[numFE].cols = r;
                numFE++;
            }

        }
    }
```

寻找兴趣值函数：

```
double FeatureExtraction::InterestValue(int c, int r)
{
    int k = w / 2;
    double F = 0;//兴趣值初始化

    double V[4] = { 0,0,0,0 };//四个方向的灰度差
    for (int i = -k; i <= k - 1; i++) {

        //水平方向
        V[0] += pow((Img.at<uchar>(c + i, r) - Img.at<uchar>(c + i + 1, r)), 2);
        //左对角线
        V[1] += pow((Img.at<uchar>(c + i, r + i) - Img.at<uchar>(c + i + 1, r + i + 1)), 2);
        //竖直方向
        V[2] += pow((Img.at<uchar>(c, r + i) - Img.at<uchar>(c, r + i + 1)), 2);
        //右对角线
        V[3] += pow((Img.at<uchar>(c + i, r - i) - Img.at<uchar>(c + i + 1, r - i - 1)), 2);

    }

    F = FindMin(V);
    return F;
}
```

三、相关系数影像匹配原理及算法

影像匹配实质上是在两幅或是多幅影像之间识别同名点。

本次影像匹配使用了基于相关系数的影像匹配算法。

基于相关系数的影像匹配算法原理：

相关系数是标准化的协方差函数，协方差函数除以两信号的方差即得相关系数。由离散灰度数据对相关系数的计算公式为：

$$\rho(c,r)=\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left(g_{i,j}\cdot g_{i+r,j+c}\right)-\frac{1}{m\cdot n}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}g_{i,j}\right)\left(\sum_{i=1}^{m}\sum_{j=1}^{n}g'_{i+r,j+c}\right)}{\sqrt{\left[\sum_{i=1}^{m}\sum_{j=1}^{n}g^{2}_{i,j}-\frac{1}{m\cdot n}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}g_{i,j}\right)^{2}\right]\left[\sum_{i=1}^{m}\sum_{j=1}^{n}g'^{2}_{i+r,j+c}-\frac{1}{m\cdot n}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}g'_{i+r,j+c}\right)^{2}\right]}}$$

编程实现：（*重点函数）

编程上使用了特征提取后的特征点进而进行影像特征匹配

```
for (int i=0;i<leftFNum;i++)
{
    loading(i, leftFNum, 2);
    Point srcPt;
    srcPt.x = leftCan[i].cols;
    srcPt.y = leftCan[i].rows;

    if (isVaildPoint(srcImgCopy, srcPt))
    {
        // 使用ROI从原图中切出一个窗口
        Rect rectSrc, rectDst;
        Mat windowSrc, windowDst;
        rectSrc = Rect(srcPt.x - r, srcPt.y - r, windowSize, windowSize);
        windowSrc = srcImgCopy(rectSrc);

        // 遍历目标图像的特征点，寻找满足条件的同名点
        double idx = 0.0, maxIdx = 0.0;
        Point maxPt;
        for (int j = 0; j < rightFNum; j++)
        {
            Point dstPt;
            dstPt.x = rightCan[j].cols;
            dstPt.y = rightCan[j].rows;
            if (isVaildPoint(dstImgCopy, dstPt))
            {
                rectDst = Rect(dstPt.x - r, dstPt.y - r, windowSize, windowSize);
                windowDst = dstImgCopy(rectDst);
                idx = computeCorrelationIdx(windowSrc, windowDst);
                if (idx > maxIdx)
                {
                    maxIdx = idx;
                    maxPt = dstPt;
                }
            }
        }
        // 判断最大的相关系数是否满足设定阈值
        if (maxIdx > threshold)
        {
            Match mat;
            mat.srcPt = srcPt;
            mat.dstPt = maxPt;
            mat.dist = maxIdx;
            matches.push_back(mat);
        }
    }
```

影像匹配函数：遍历两个影像特征点数组，每个特征点处剪出一个窗口进行逐一求取相关系数，大于阈值则为同名点

计算两个相同大小的窗口相关系数：

```
double ImageMatching::computeCorrelationIdx(Mat& srcWindow, Mat& dstWindow)
{
    if (srcWindow.size != dstWindow.size)
    {
        cerr << "窗口大小不匹配！" << std::endl;
        return 0;
    }

    // 总像元数量
    double totalNum = srcWindow.rows * srcWindow.cols;

    // 计算两张影像窗口中像素的平均值
    double gAverSrc = 0.0, gAverDst = 0.0;
    for (int i = 0; i < srcWindow.rows; i++)
        for (int j = 0; j < srcWindow.cols; j++)
        {
            gAverSrc += srcWindow.at<unsigned char>(i, j);
            gAverDst += dstWindow.at<unsigned char>(i, j);
        }
    gAverSrc /= totalNum;
    gAverDst /= totalNum;

    // 计算相关系数
    double numerator = 0.0;
    double denominatorSrc = 0.0;
    double denominatorDst = 0.0;

    for (int i = 0; i < srcWindow.rows; i++)
        for (int j = 0; j < srcWindow.cols; j++)
        {
            numerator += (srcWindow.at<unsigned char>(i, j) - gAverSrc) * (dstWindow.at<unsigned char>(i, j) - gAverDst);
            denominatorSrc += pow((srcWindow.at<unsigned char>(i, j) - gAverSrc), 2);
            denominatorDst += pow((dstWindow.at<unsigned char>(i, j) - gAverDst), 2);
        }

    double denominator = sqrt(denominatorSrc * denominatorDst);
    return abs(numerator / denominator);
}
```

四、编程思路以及流程图

本次编程利用面向对象的设计，设计了一个图像处理类。

父类：ImageSolution（图像处理类）

```
class ImageSolution
{
public:
    Mat leftImg, rightImg;//左图右图
    ImageSolution();
    void ReadImg();//读取图像
    ~ImageSolution();
};
```

子类：FeatureExtraction（特征提取类）

并建立一个结构存储特征点的位置以及兴趣值。

```
//像素特征值
struct FeaturePoint {
    double value;
    int rows;
    int cols;
};

#define w 2 //兴趣点搜索窗口大小
#define corner_threshol 700
class FeatureExtraction :
    public ImageSolution
{
public:

    int c_threshol= corner_threshol;
    Mat Img, FeatureImg;//特征提取后的影像
    FeaturePoint* Candidate;//存储特征点
    int numFE = 0;//记录兴趣点（特征点）数量
    FeatureExtraction();
    FeatureExtraction(Mat Img)
    {
        this->Img = Img;
        cvtColor(Img, FeatureImg, COLOR_GRAY2RGB);
    };

    void SetThreshol(int threshol);
    void GetFuature();//特征提取运算
    double InterestValue(int c, int r);//计算窗口像素四方向灰度差
    double FindMin(double V[4]);//找出最小值作为兴趣值

    void ShowFeature();//展示特征点
    void WriteFeature(const char*filename);//写一个txt文件存储特征点
    ~FeatureExtraction();
};
```

ImageMatching(影像匹配类-继承特征提取类)

```
//同名点结构
struct Match
{
    cv::Point2d srcPt; // 左片像点坐标
    cv::Point2d dstPt; // 右片像点坐标
    double dist;       // 相似性测度的计算值
};
#define corr_window_size 25    // 预设 相关系数匹配窗口大小
#define corr_threshold  0.85 // 预设 相关系数匹配阈值
class ImageMatching :
    public FeatureExtraction
{
public:
    Mat outputImg;//输出图像
    int windowSize;//窗口大小
    double  threshold;//阈值

    FeatureExtraction left,right;//左、右特征点对象
    FeaturePoint* leftCan, * rightCan;//存储左右两图的特征点
    int leftFNum, rightFNum;//特征点数量
    vector<Match> matches;//存储同名点

    ImageMatching();
    ImageMatching(FeatureExtraction L, FeatureExtraction R)
    {
        leftImg = L.Img;
        rightImg = R.Img;
        leftCan = L.Candidate;
        leftFNum = L.numFE;
        rightCan = R.Candidate;
        rightFNum = R.numFE;

    }
    void setWindowSize(int WindowSize);//设置窗口大小
    void setThreshold(double threshold);//设置

    void GGetFeature();//未初始化左图右图特征点对象 则进行特征提取
    void FileReadCan(const char* filename);//文件读取特征点(预先读取好)

    void Matching();//匹配
    bool isVaildPoint(Mat M,Point &pt);//判断是否是窗口点
    double computeCorrelationIdx(Mat& srcWindow,Mat& dstWindow);//计算两窗口相关系数

    void drawMatches();//绘制影像匹配窗口
};
```

流程图：

main 函数）

```
tureExtractionAndMatching                                    (全局范围)
int main()
{
    cv::utils::logging::setLogLevel(utils::logging::LOG_LEVEL_SILENT);//不再输出opencv日志

    ImageSolution M;
    M.ReadImg();

    TicToc t = TicToc();

    t.tic();
    FeatureExtraction T1(M.leftImg);
    T1.SetThreshol(corner_threshol); T1.MoravecGetFuature();
    double time=t.toc();
    cout << "左图使用Moravec算子进行特征提取，阈值设置为" << T1.c_threshol<<endl;
    cout << "特征点有" << T1.numFE<<"个" << endl;
    cout << "用时：" << time << "秒\n" << endl;

    t.tic();
    FeatureExtraction T2(M.rightImg);
    T2.SetThreshol(corner_threshol); T2.MoravecGetFuature();
    time = t.toc();
    cout << "右图使用Moravec算子进行特征提取，阈值设置为" << T2.c_threshol << endl;
    cout << "特征点有" << T2.numFE << "个" << endl;
    cout << "用时：" << time << "秒\n" << endl;

    T1.WriteFeature("左图特征点.txt");//只输出特征点
    T2.WriteFeature("右图特征点.txt");
    imshow("左图特征提取后",T1.FeatureImg);
    imshow("右图特征提取后", T2.FeatureImg);
    waitKey(0);
    imwrite("左图特征提取.jpg", T1.FeatureImg);
    imwrite("右图特征提取.jpg", T2.FeatureImg);

    t.tic();
    ImageMatching K(T1, T2);
    K.setWindowSize(corr_window_size);
    K.setThreshold(corr_threshold);
    K.Matching();
    K.drawMatches();
    time = t.toc();
    cout << "使用基于相关的影像匹配，窗口设置大小为" << K.windowSize << "\t阈值为" << K.threshold << endl;
    cout << "相关系数匹配到同名点：" << K.matches.size() << endl;
    cout << "用时" << time << "秒\n";
    imshow("基于相关系数的影像匹配", K.outputImg);
    waitKey(0);
    imwrite("基于相关系数的影像匹配.jpg", K.outputImg);
    return 0;

}
```

流程：

1. ImageSolution：

1.1. 读取文件

```
void ImageSolution::ReadImg()
{

    leftImg = imread("./RawImage/u0369_panLeft.tif", 0);
    rightImg = imread("./RawImage/u0367_panRight.tif", 0);
    imshow("RawImageLeft", leftImg);
    imshow("RawImageRight", rightImg);
    waitKey(0);
}
```

2. FeatureExtration:

2.1. 两张图像进行 **Moravec** 特征点提取，并存储到特征点结构的数组里（略）（函数如**第二部分**所示）

2.2. 特征点提取后进行绘制，先将图像化为彩色图，再在像素周围绘制十字架）

```
void FeatureExtraction::ShowFeature()
{
    int imgRows = FeatureImg.rows;
    int imgCols = FeatureImg.cols;
    for (int k = 0; k < numFE; k++)
    {

        loading(k, numFE,1);
    /*  cout << k << "\t" << Candidate[k].rows << "\t" << Candidate[k].cols << endl;*/
        if (Candidate[k].value != 0 )
        {
            for (int p = -3; p < 7; p++)
            {
                if (Candidate[k].rows + 6 < imgRows && Candidate[k].rows - 3 > 0 && Candidate[k].cols + 6 < imgCols && Candidate[k].cols - 3 > 0)
                { //画红十字（周围像素变红色）
                    FeatureImg.at<Vec3b>(Candidate[k].rows + p, Candidate[k].cols + p)[0] = 0;
                    FeatureImg.at<Vec3b>(Candidate[k].rows + p, Candidate[k].cols + p)[1] = 0;
                    FeatureImg.at<Vec3b>(Candidate[k].rows + p, Candidate[k].cols + p)[2] = 255;

                    FeatureImg.at<Vec3b>(Candidate[k].rows + p, Candidate[k].cols - p)[0] = 0;
                    FeatureImg.at<Vec3b>(Candidate[k].rows + p, Candidate[k].cols - p)[1] = 0;
                    FeatureImg.at<Vec3b>(Candidate[k].rows + p, Candidate[k].cols - p)[2] = 255;
                }
            /*  Img.at<uchar>(Candidate[k].rows + p, Candidate[k].cols + p) = 0;
                Img.at<uchar>(Candidate[k].rows + p, Candidate[k].cols - p) = 0;*/

            }
        }
    }


}
```

特征点的文件输出

```
void FeatureExtraction::WriteFeature(const char* filename)
{

    FILE *fp;
    errno_t error;
    error=fopen_s(&fp, filename, "w");
    if (error != 0)
    {
        cout << "文件打开失败";
    }

    fprintf_s(fp, "%d\n", numFE);
    fprintf_s(fp, "num\tValue\trows\tcols\t\n");
    for (int i = 0; i < numFE; i++)
    {
        fprintf_s(fp, "%d\t%lf\t%d\t%d\t\n", i, Candidate[i].value, Candidate[i].rows, Candidate[i].cols);
    }

    fclose(fp);
}
```

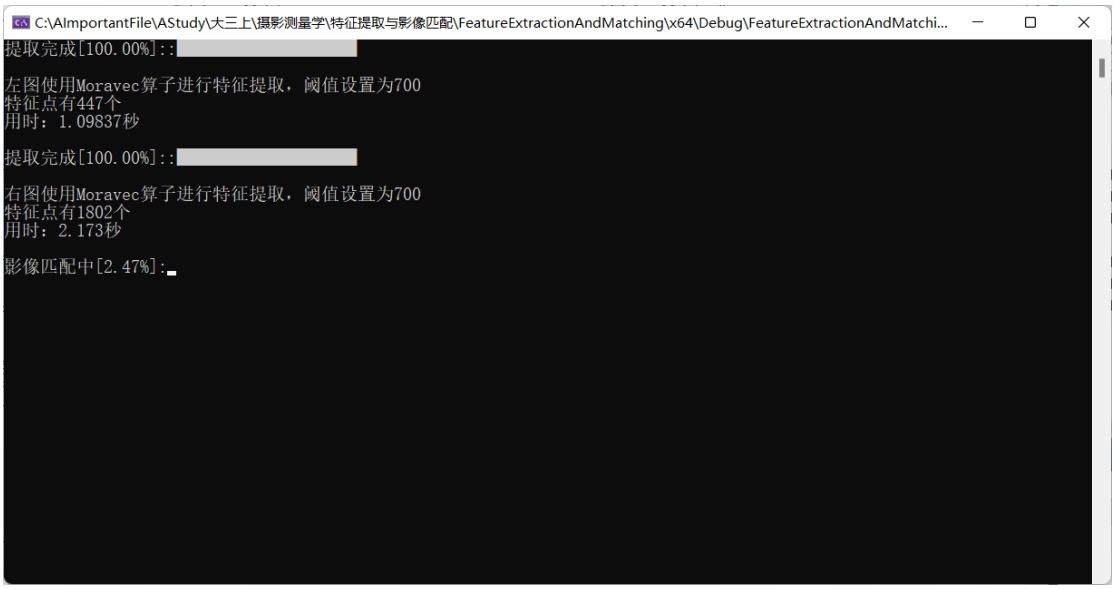
3. ImageMatching:

两张图根据特征点的提取进行基于相关系数的影像匹配，寻找同名点并存储在同名点结构的数组里（略）（函数如**第三部分**所示）

3.2. 创建新的窗口进行影像匹配的窗口展示，并绘制同名点连线。（彩色图）

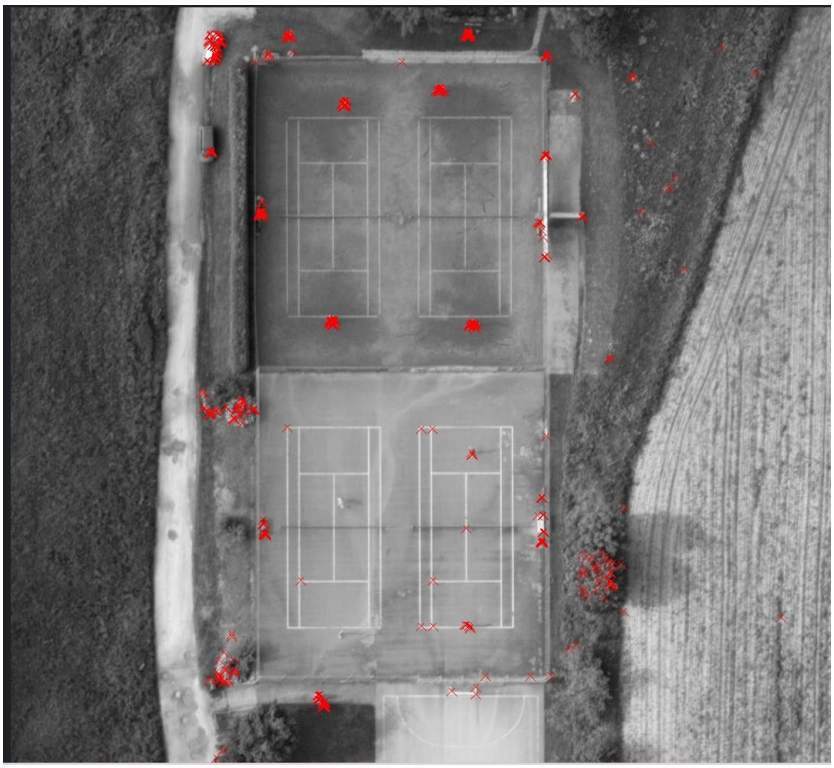
```
void ImageMatching::drawMatches()
{
    cvtColor(leftImg, leftImg, COLOR_GRAY2RGB);
    cvtColor(rightImg, rightImg, COLOR_GRAY2RGB);
    outputImg.create(Size(leftImg.cols + rightImg.cols, std::max(leftImg.rows, rightImg.rows)), CV_8UC3);

    leftImg.copyTo(outputImg(Rect(0, 0, leftImg.cols, leftImg.rows)));
    rightImg.copyTo(outputImg(Rect(leftImg.cols, 0, rightImg.cols, rightImg.rows)));

    Point pt1, pt2;

    for (Match match : matches)
    {
        Scalar color(0,0,255);

        pt1 = match.srcPt;
        pt2 = Point(match.dstPt.x + leftImg.cols, match.dstPt.y);

        circle(outputImg, pt1, 5, color, 1);
        circle(outputImg, pt2, 5, color, 1);
        line(outputImg, pt1, pt2, color, 1);
    }
}
```

设置了进度条输出：

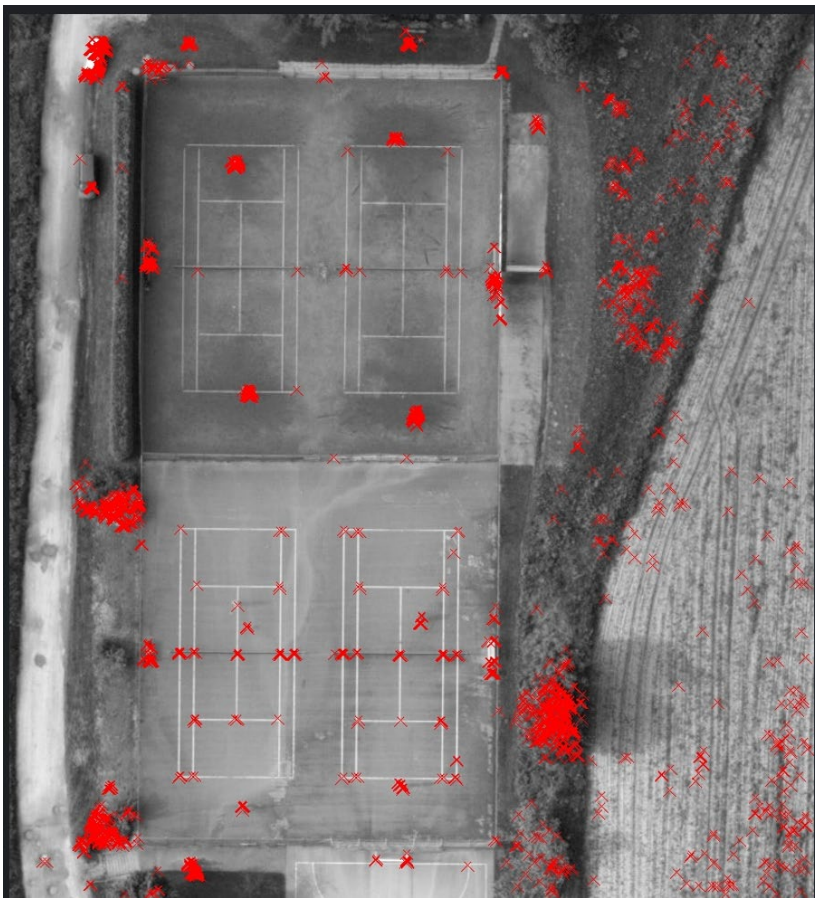


五、实验结果分析

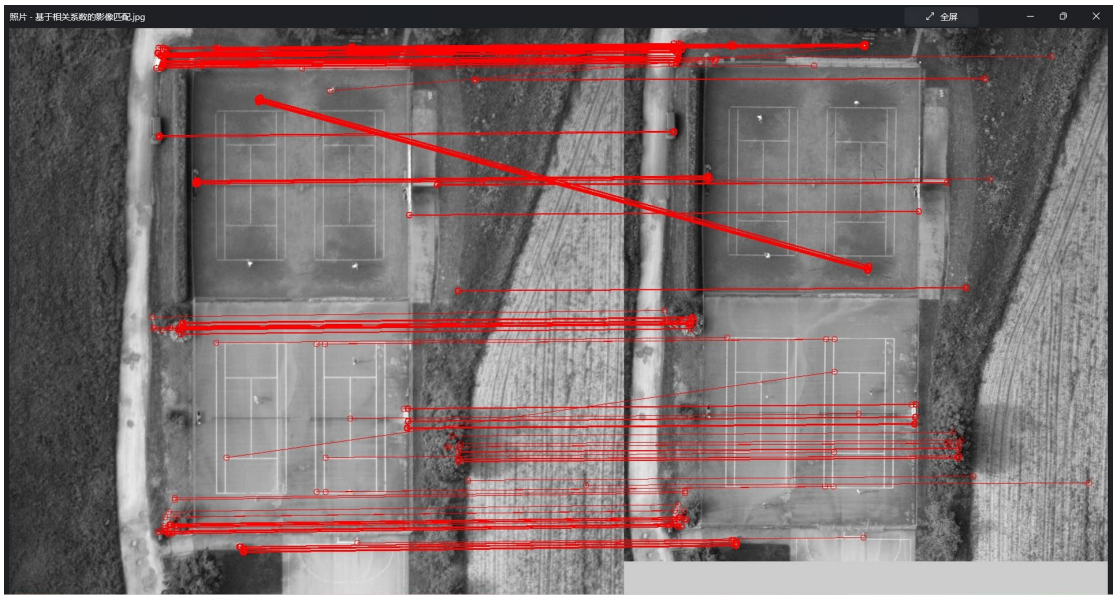
1. 特征提取阈值：700
影像匹配窗口大小：25 阈值：0.85



.jpg（阈值：700）



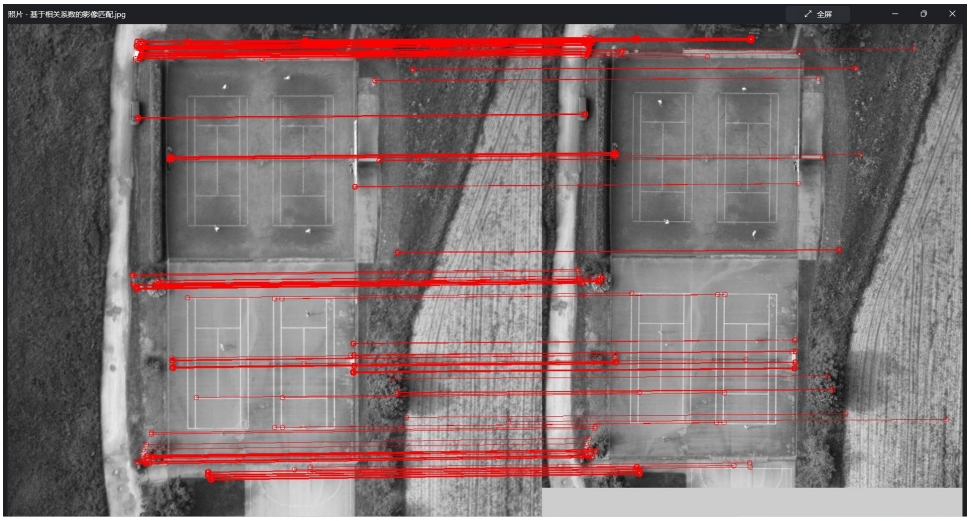
.jpg（阈值：700）



.jpg(窗口大小：25 阈值：0.85)

2. 特征提取阈值：700
影像匹配窗口大小：50 阈值：0.85

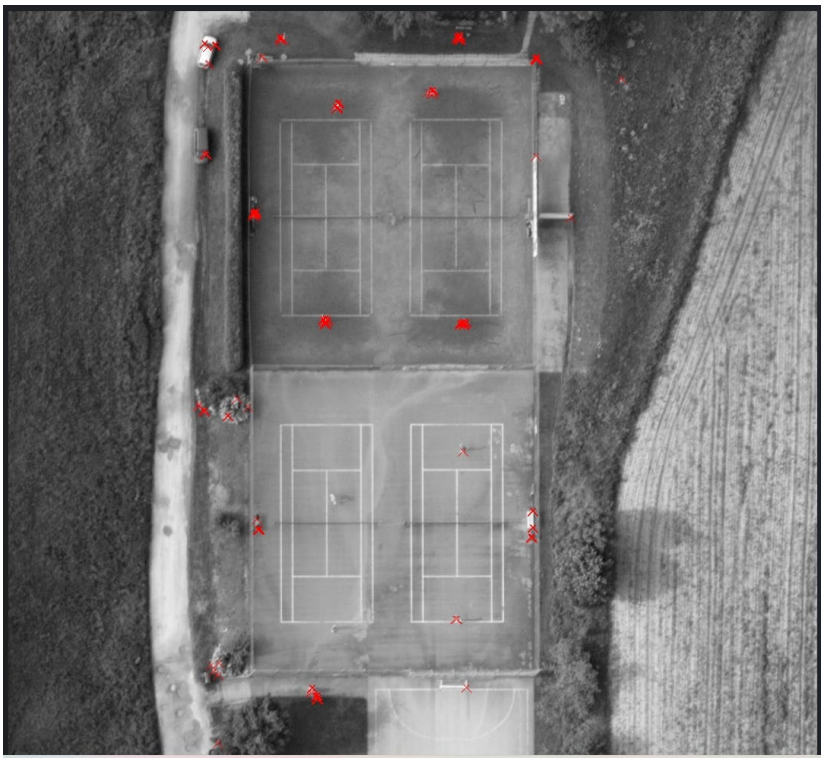
影像匹配窗口变大， 可见匹配的同名点变多，但准确度变高



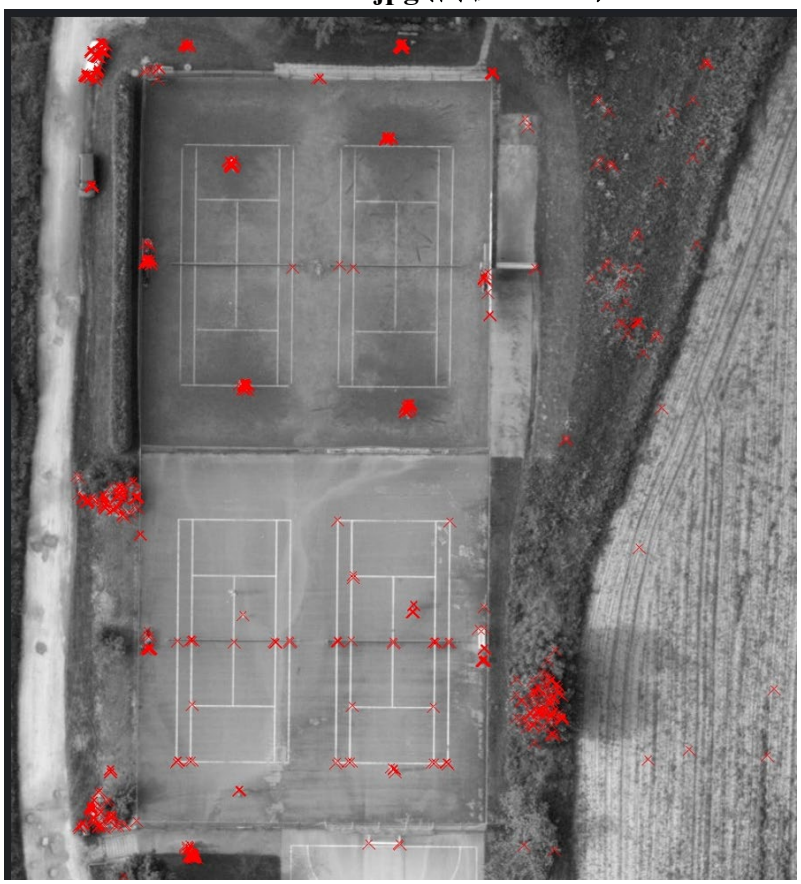
.jpg(窗口大小：50 阈值：0.85)

3. 特征提取阈值：1200

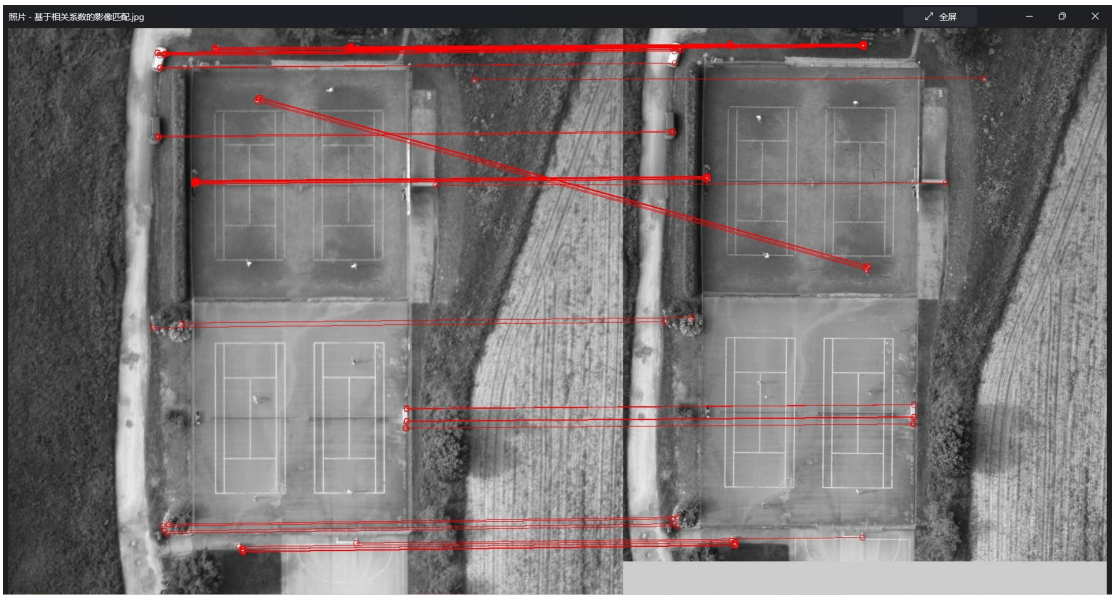
影像匹配窗口大小：25 阈值：0.85



.jpg(阈值：1200)



.jpg(阈值：1200)



.jpg(窗口大小：25 阈值：0.85)
（特征提取阈值：1200）