CENTRAL SOUTH UNIVERSITY

面向对象设计课程报告

基于 SIFT 算法的特征点图像拼接

学号: 8201210220
姓名: 彭奇
班级: 信息与计算科学 2101
导师: 刘圣军

September 13, 2023

Contents

1 问题描述

Question 1.1. 由于相机的拍摄范围有限，当想要把一个超大的场景拍摄进一张图片，通常情况下是拍摄多张图片，然后进行图像拼接。

Requirment 1.2. 以下三点

- 基本软件界面设计与绘制生成，包括三个图像显示区域，两张原始图片及一张拼接后图片；
- 两张原始图片特征点的检测；生成特征点对应；生成一张图片的平移旋转矩阵，并对其进行相应操作，拼合图片；
- 图片文件输入输出。

2 问题分析

经学习图像拼接问题可以大致按如下流程行进。按分期可分为前期工作（特征点提取和配位），中期（投影变换）, 后期（图像拼合）。其中投影变换的数学算法转换以及后期图像复制再拼合相对较难。

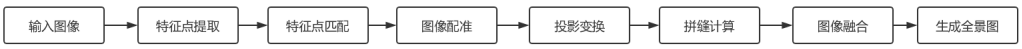

Figure 1: 流程图

2.1 特征提取算法

特征提取（Feature extraction）在机器学习、模式识别和图像处理中有很多的应用。特征提取是从一个初始测量的资料集合中开始做，然后建构出富含资讯性而且不冗余的导出值，称为特征值（feature）。

当一个算法的输入资料太过于庞大冗余以至于不便处理（如：一样的测量方法但是分别使用英尺和米表示，或是影像中像素的重复性），这些资料可以被转换成化简后的特征集合，也称作特征向量（feature vector）

2.2 透视变换

透视变换（Perspective Transformation）是一种在计算机视觉和图像处理领域中常用的技术，用于将一个平面上的二维图像映射到另一个平面上，同时考虑到透视效果（远近物体的大小和角度变化）。这种变换通常用于图像矫正、图像重建、增强现实、虚拟现实和三维重建等应用。

透视变换的主要目标是根据不同视角观察物体时产生的透视效果，将原始图像的像素映射到新的平面上，使得在新的平面上的图像看起来更加真实或者适合特定的应

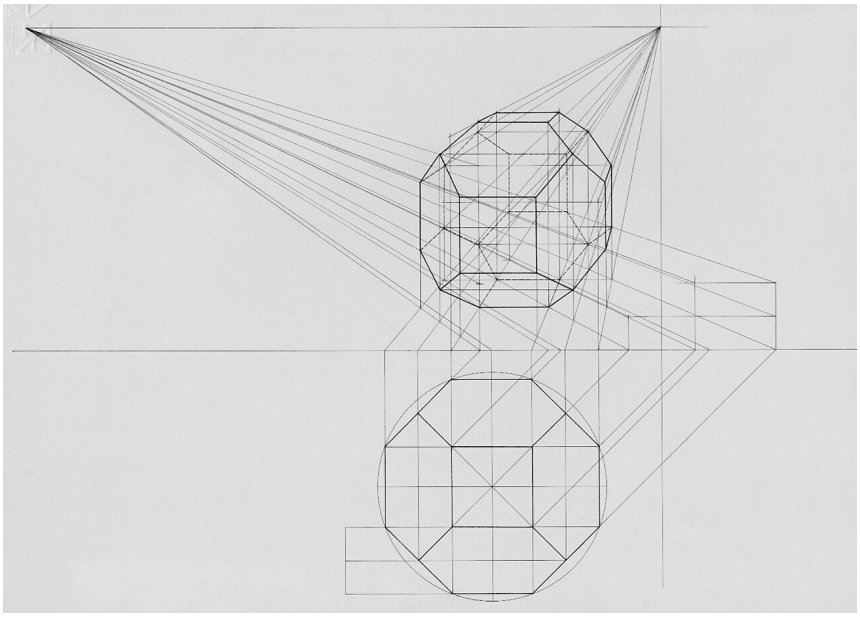

Figure 2: 透视图

用场景。这通常涉及到定义一个变换矩阵，该矩阵描述了如何将原始图像中的点映射到新的位置。

透视变换的步骤通常包括以下几个方面 [2]：

1. 定义原始图像上的四个关键点，这些点通常形成一个矩形或平行四边形，表示原始图像上的感兴趣区域（ROI）。

2. 定义新的目标平面上的对应四个点，这些点通常构成一个矩形，表示希望将原始图像映射到的目标区域。

3. 计算透视变换矩阵，它将原始图像上的点映射到目标平面上的点。

4. 应用透视变换矩阵，将原始图像进行变换，得到透视校正后的图像。

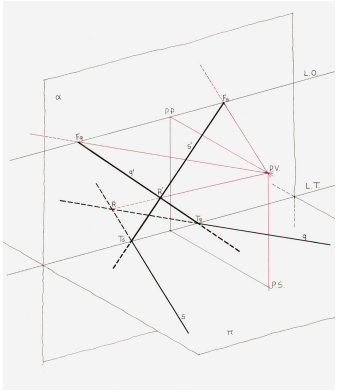

Figure 3: 透视变换

2.3 图像拷贝

在 OpenCV 中，拷贝图像数据时有两种方式：深拷贝（Deep Copy）和浅拷贝（Shallow Copy）。这两种拷贝方式的主要区别在于是否创建新的图像副本。

浅拷贝（Shallow Copy）是指将图像对象的指针复制给另一个对象，这两个对象将共享相同的图像数据。当对其中一个图像进行修改时，另一个图像也会受到影响。这是因为只复制了指针，而没有创建新的图像数据。

深拷贝（Deep Copy）是指创建一个新的图像对象，并复制原始图像的像素数据到新对象中。这样每个图像对象都拥有自己的独立像素数据，修改一个图像不会影响其他图像。

深拷贝函数 copyTo()[1].

3 设计与实现

3.1 提前准备 (9.3)

- 根据要求搭建 Opencv 环境：在 VC 中的属性管理器添加相关路径。
- MFC 绘制符合要求的 Dialog：包括显示两个原始图片和拼接图片的三个 picture control 控件，并定义好相对应的功能按钮，对 ID 进行规范设置。

3.2 窗口设计 (9/3-9/4)

- 实现根据文件对话框的读取文件的功能。
- 将选取路径的图片根据相适应控件大小进行重采样，展示到控件上。(难点，耗时较长)[6].

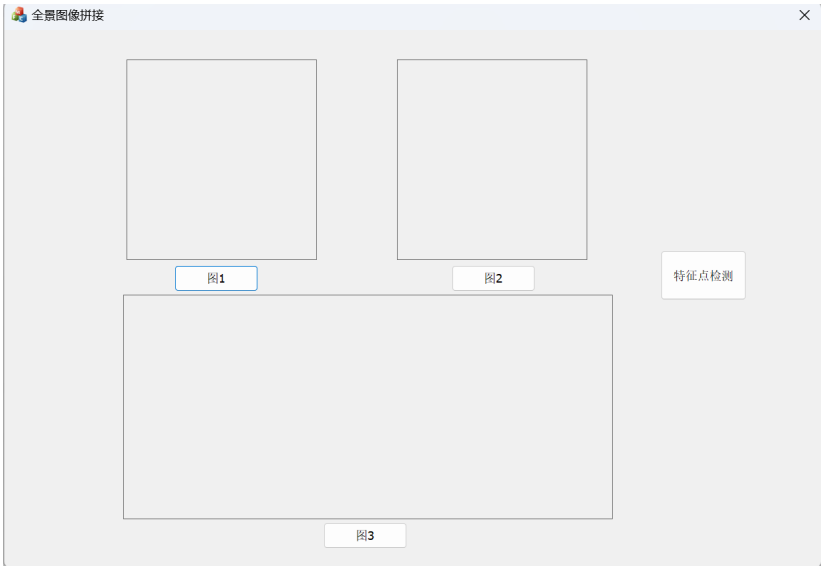

Figure 4: 界面设计

3.3 正式工作 (9/5-9/9)

- 特征点提取和匹配

Ways 1. SIFT （Scale-invariant feature transform）
SIFT 算法，又称尺度不变特征转换算法，是一种图像局部特征提取算法，它通过在不同的尺度空间中寻找极值点（特征点，关键点）的精确定位和主方向，构建关键点描述符来提取特征。SIFT 提取的关键点具有尺度不变性、旋转不变性，而且不会因光照、仿射变换和噪音等因素而干扰。SIFT 所查找到的关键点是一些十分突出、不会因光照、仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等 [3]。

- 定义特征检测器（SIFT,SURF,ORB 等）。
- 对图像中特征点进行检测，并将特征点存储在 Keypoints 中。
- 提取特征点的描述信息。
- 定义特征匹配器（主要有两种分别为暴力匹配 BFmatch 和 FlannBased）。
- 筛选出优秀的特征点（一般根据临近两点的距离进行过滤，根据 DMatch 中的 distance 作阈值，distance 越小说明检测点的相似度越高）
- 连接并显示匹配的特征点 [4]

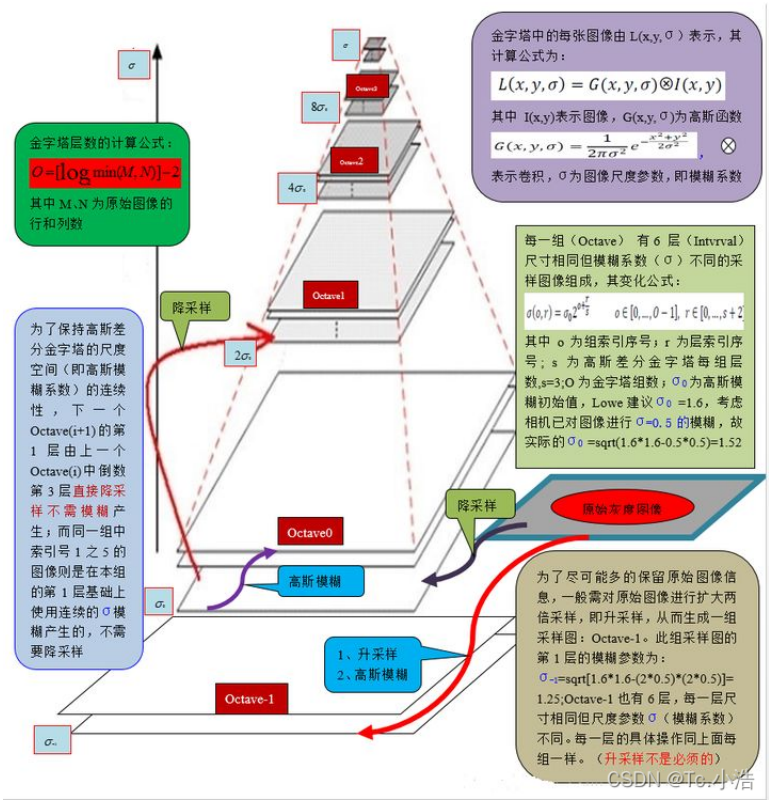

Figure 5: GAUSS 金字塔

Code 1.

```
//定义变量
vector<KeyPoint> keypoints1, keypoints2;
Mat descriptors1, descriptors2;

//创建sift特征检测器,并计算出特征向量
Ptr<SIFT>detector = SIFT::create();
detector->detectAndCompute(left, Mat(),
keypoints1, descriptors1);
detector->detectAndCompute(right, Mat(),
keypoints2, descriptors2);

//创建匹配器为暴力匹配器,并进行匹配
Ptr<DescriptorMatcher> matcher =
DescriptorMatcher::create
(DescriptorMatcher::BRUTEFORCE);
vector<DMatch> matches;
vector<Mat>train_desc(1, descriptors2);
matcher->add(train_desc);
matcher->train();

//筛选出良好特征点
vector<vector<DMatch>> matchpoints;
matcher->knnMatch(descriptors1, matchpoints, 2);
vector<DMatch> goodfeature;
for (int i = 0; i < matchpoints.size(); i++)
{
    if (matchpoints[i][0].distance <
    0.15 * matchpoints[i][1].distance)
    {
        goodfeature.push_back(matchpoints[i][0]);
    }
}

//输出关键点和匹配结果
Mat result_img;
drawMatches(left, keypoints1, right,
keypoints2, goodfeature, result_img);
drawKeypoints(left, keypoints1, left);
drawKeypoints(right, keypoints2, right);
```

- 透视变换

Ways 2. 透视变换（Perspective Transformation）的本质是将图像投影到一个新的视平面，其通用变换公式为：$X = AX'$,X 为原始图像像素点的齐次坐标,X' 为变换之后的图像像素点的齐次坐标。
原始图像像素点的齐次坐标 $x'\ y'\ z'$ 对应变换之后的图像像素点的齐次坐标 $X;Y;Z$ 。将 Z 归一化后，得到归一化后的齐次坐标（X';Y';1），则其变换之后的图像像素点的齐次坐标（$X;Y;Z$）对应的二维平面坐标为（$X';Y'$）即当齐次坐标 $Z' = 1$，则点（$X';Y'$）就是原图像对应的像素点的二维平面坐标经过变换后的二维平面坐标。
在这个方程中，总共有 8 个未知数，因此我们可以找到 4 对 8 个点（每个两个对应，原平面上的点和其对应映射到新的视平面的点为一对）列出方程。解出参数 $a_{11}\ a_{12}\ a_{13}\ a_{21}\ a_{22}\ a_{23}\ a_{31}\ a_{32}$。假设这 8 个点为：
- 源点四个坐标分别为 $A\ (x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$
- 目标点四个坐标分别为 $B\ (X'_0, Y'_0), (X'_1, Y'_1), (X'_2, Y'_2), (X'_3, Y'_3)$

则带入方程（写成矩阵形式）即可。

Code 2.

```
    //透视转换,得到拼接图片
    Mat homo = findHomography(imagepoint1,
 imagepoint2, RANSAC);
    Mat result_image;
    CalcCorners(homo, right);
    warpPerspective(right, result_image, homo,
Size(MAX(corners.right_top.x,
corners.right_bottom.x), left.rows));
```

- 图像拷贝及拼接

Code 3.

```
    int dp_width = result_image.cols;
    int dp_height = result_image.rows;
    Mat dp(dp_height, dp_width, CV_8UC3);
    dp.setTo(0);
    result_image.copyTo(dp(Rect(0, 0,
    result_image.cols, result_image.rows)));
    left.copyTo(dp(Rect(0, 0, left.cols, left.rows)));
```

3.4 拓展工作 (9/10-9/11)

- 消除拼接处的分离感。

处理思路是加权融合，图像的重叠区域的像素值按一定的权值相加合成新的图像。/参考自网络 [5]。

Code 4.

```
void OptimizeSeam(Mat& img1, Mat& trans, Mat& dst)
  {
     int start = MIN(corners.left_top.x,
     corners.left_bottom.x);
     double processWidth = img1.cols - start;
     double alpha = 1;//img1中像素的权重,初始设为1
     for (int i = 0; i < dst.rows; i++)
     {
      uchar* p = img1.ptr<uchar>(i);
      uchar* t = trans.ptr<uchar>(i);
      uchar* d = dst.ptr<uchar>(i);
      for (int j = start; j < img1.cols; j++)
      {
        //如遇图像trans中无像素的黑点，则完全拷贝img1
        if (t[j * 3] == 0 && t[j * 3 + 1] ==
        0 && t[j * 3 + 2] == 0)
        {
         alpha = 1;
        }
        else
        {
        //img1中像素的权重，与当前处理点距重叠区域左边界
        的距离成正比
         alpha =
         (processWidth - (j - start)) / processWidth;
        }
         d[j * 3] =
         p[j * 3]*alpha + t[j * 3]*(1 - alpha);
         d[j * 3 + 1] =
         p[j * 3 + 1]*alpha + t[j * 3 + 1]*(1 - alpha);
         d[j * 3 + 2] =
         p[j * 3 + 2]*alpha + t[j * 3 + 2]*(1 - alpha);
      }
    }
}
```

3.5 报告撰写 (9/12-9/13)

按照模板要求，根据代码和思路历程完成报告撰写。

4 结果与讨论

4.1 成果展示

- 点击打开文件管理器 [?]

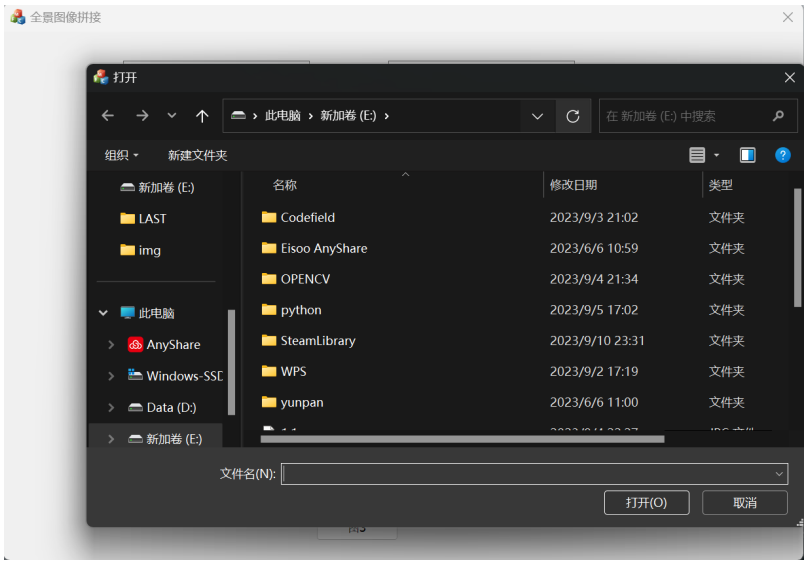

Figure 6: 连接并显示匹配的特征点

- 插入图片

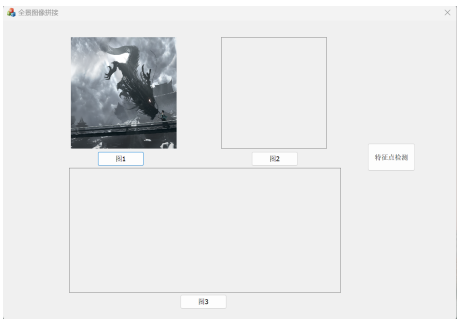

Figure 7: 插入左图

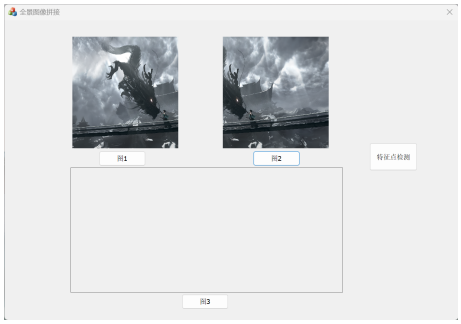

Figure 8: 插入右图

- 点击特征点检测
- 拼接图像

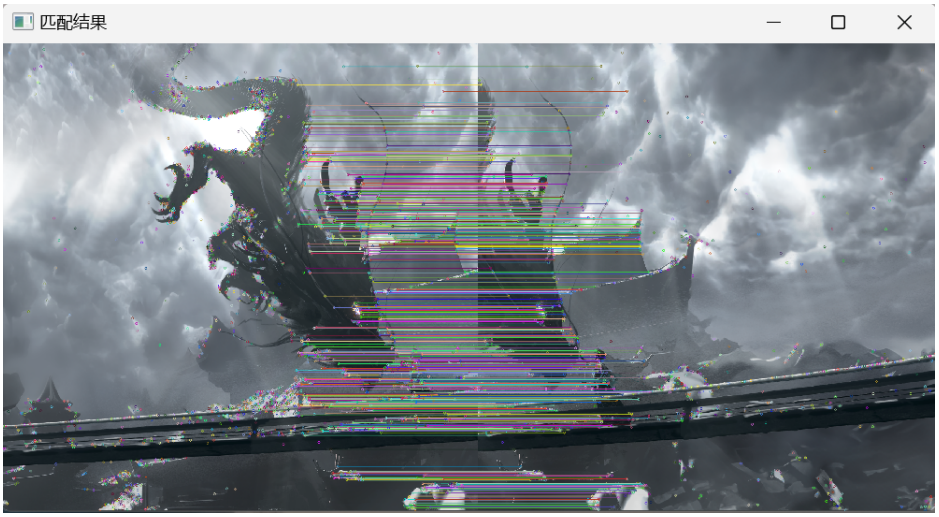

Figure 9: 连接并显示匹配的特征点

Figure 10: 全景图展示

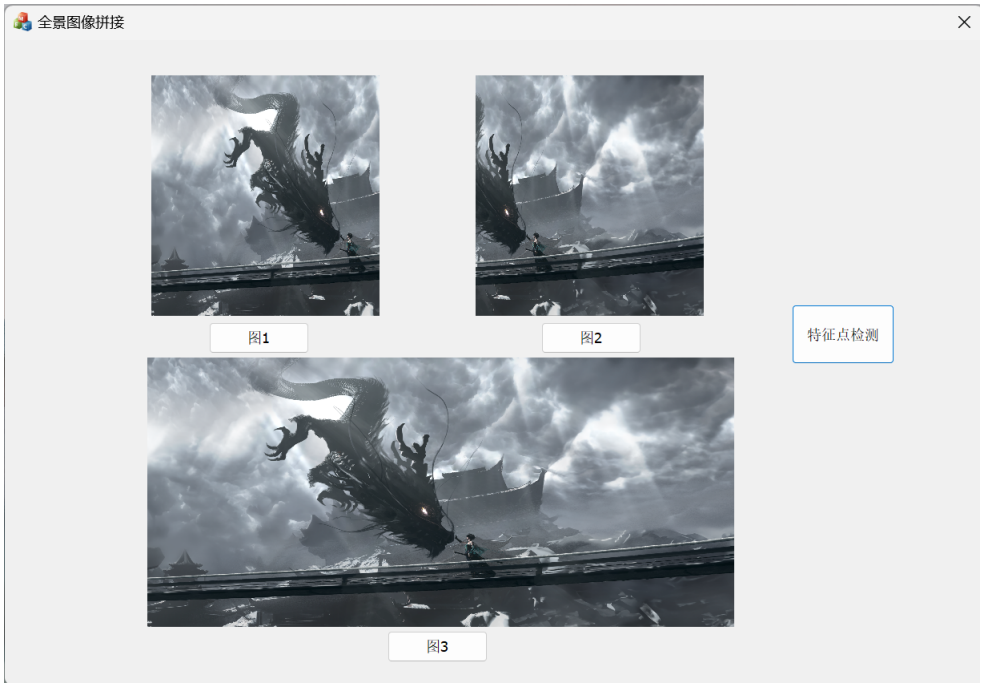

Figure 11: 界面体现

4.2 思路历程

1. 需求分析：

• 首先，我明确了项目的主要目标：基于 MFC 开发一款实现图像拼接的软件，以将两张原始图片拼接成一张全景图像。我们确定了基本要求，包括界面设计、特征点检测、图像拼接和文件输入输出。

2. 选择库：

• 为了实现图像处理功能，经过资料查询，我决定选择了 OpenCV 作为图像处理库。OpenCV 提供了丰富的图像处理和计算机视觉功能，非常适合我们的项目。

3. 界面设计：

• 基于要求我开始设计软件的用户界面。创建基于对话框的 MFC 窗口，我添加了三个图像显示区域，可以实现文件对话框选择按钮以及拼接和显示特征点的按钮。

4. 特征点检测和匹配：

• 我选择了 SIFT 算法来检测和描述特征点，并使用暴力匹配器进行特征点匹配。SIFT 相对 SURF 和 ORB 性能相对优越，适合用于全景图像拼接。

• 在特征点匹配过程中，我们引入了一些筛选机制，以筛选出良好的特征点。

5. 计算平移旋转矩 *：

• 使用匹配的特征点对，我们计算了透视矩阵，该矩阵将用于将一张图片映射到另一张图片的坐标空间中。

6. 图像拼接：

• 我们使用生成的矩阵，将一张图片映射到另一张图片上，然后进行图像融合操作，以生成全景图像。

• 图像融合采用了权重平均的方法，确保拼接边缘自然过渡。

7. 测试与调试：

• 我们对软件进行了广泛的测试，使用不同场景和图片进行验证。我们修复了一些细节问题。

8. 结果讨论

• 经逐项检验，基本满足了题目中的各项要求。代码完整，有一定注释。实验相对成功。

4.3 问题汇总

Trouble 4.1. 安装 Opencv 后仍然无法使用相关头文件

Solution 4.1. 在 VC 属性管理器中配置相关路径。

Trouble 4.2. Opencv 配置无法在别的工程下使用

Solution 4.2. 将配置路径生成的 props 文件保存，在其他工程下添加该文件即可。

Trouble 4.3. 文件对话框打开的文件不能直接使用

Solution 4.3. 文件对话框路径获取：GetPathName 函数返回的是 CString 类型，而 imread() 只能读取 String 类。经查询可以利用 CW2A 进行转换。

Trouble 4.4. 无法创建 SURF，ORB 的特征检测器

Solution 4.4. Opencv4 版本中，拼接图像所用到的特征点提取的部分算法（Surf 提取法，Orb 提取法等）封装在 <stitching/stitcher.hpp> 头文件中，从官网的库中可以下载到这个头文件。但 SIFT 算法并没有被封装，也可以使用 SIFT 算法。

Trouble 4.5. 图像拷贝问题

Solution 4.5. 无论是读取图像展示到控件的复制，还是后期图像拼接进行的拷贝。都可以利用 CopyTo() 函数深拷贝。

Trouble 4.6. 如何进行透视变换

Solution 4.6. 利用计算出的已有的透视矩阵，”warpPerspective” 函数可以根据指定的参数执行透视变换。

5 总结

完成该实验，对计算机图形学有了初步了解。认识到了这是一门强大的武器，提供了诸多方法。开阔了视野。意识到了很多常见的问题也与图形学相关。

对 C++ 这门语言结构认识的更深刻了，在反复的 Debug 中知道了一些常见报错的原因，这也加深了我的理解。

在这个实践中，反复查询资料，发现相当多数学知识的运用，本质上虽然简单质朴，但却能合力解决一个实际的难题。

在未来的日子，也会继续深入对数学的学习，更实际地了解编程语言的运用。

References

[1] Hdnw, "Opencv(十五)：拷贝图像，csdn," 2022.

[2] 淡定的炮仗, "计算机视觉（十四）透视变换，csdn," 2021.

[3] Tc. 小浩, "全网最详细 sift 算法原理实现，csdn," 2022.

[4] crossoverpptx, "Sift 算法，csdn," 2022.

[5] 牧野, "Opencv sift 和 surf 特征实现图像无缝拼接生成全景图像，csdn," 2016.

[6] 鸡啄米, "Vs2010/mfc 编程入门之十七（对话框：文件对话框）," 2010.