摘　要

随着人工智能和物联网技术的不断发展，自主控制和智能化应用已成为人们生活和产业发展的重要趋势。视觉定位系统作为其中的关键支撑技术之一，其对于实现无人机、自动驾驶等智能设备的精确定位、路径规划等方面至关重要，因此受到了越来越多的关注和研究。

本文旨在研究视觉定位系统中基于图像特征点的技术难点和解决方法，以为其未来的发展提供有力的技术支持。首先，本文对图像特征点提取算法进行了分析，并探讨了SIFT、SURF、ORB、AKAZE 等图像特征提取算法的特点和应用。接着，本文设计了一套基于树莓派 4B、UPS 电路和 USB 双目摄像头的图像采集系统，并采用 Flask 和 OpenCV 搭建图传服务器，通过 Node-RED 物联网系统实现数据传输。此外，本文还开发了一个 Windows 应用程序，利用 OpenCV 和 PySide6 等开源库进行图像处理，并通过 FLANN 匹配器和 RANSAC 算法进行图像特征点的匹配与定位。为满足多场景需求，本文还采用了 SuperPoint 和 SuperGlue 算法以提高系统的精度。

实验结果显示，本系统在全局和局部图片的特征点提取方面表现良好。水平偏移角和垂直偏移角的误差控制在 5%以内，直线距离的误差也控制在 5%以内。最后，对多种算法的运行效果进行排序，以获得最小的误差值，为未来实际应用提供了有力支持。

关键词：视觉定位；特征提取算法；特征匹配算法；UPS；OpenCV

Abstract

With the continuous development of artificial intelligence and the Internet of Things technology, autonomous control and intelligent applications have become important trends in people's lives and industrial development. As one of the key supporting technologies, visual localization system plays a vital role in achieving accurate positioning and path planning for intelligent devices such as drones and autonomous driving. Therefore, it has received more and more attention and research.

This paper aims to explore the technical challenges and solutions of visual localization system based on image feature points in practical applications, and provide strong technical support for its future development. Firstly, the image feature point extraction algorithm is studied, and the characteristics and applications of SIFT, SURF, ORB, AKAZE and other image feature extraction algorithms are analyzed. Secondly, an image acquisition system based on Raspberry Pi 4B, UPS circuit and USB binocular camera is designed, and a video transmission server is built using Flask and OpenCV, and data transmission is realized through the Node-RED IoT system. In the image processing part, a Windows application is developed using open source libraries such as OpenCV and PySide6, and the FLANN matcher and RANSAC algorithm are used for image feature point matching and localization. Furthermore, the SuperPoint and SuperGlue algorithms are used to improve the accuracy of the system.

Experimental results show that the system performs well in feature point extraction of global and local images. The errors of horizontal and vertical offset angles are controlled within 5%, and the error of straight-line distance is also controlled within 5%. Finally, the running effects of multiple algorithms are ranked to obtain the minimum error value, which provides strong support for future practical applications.

Key words： Visual localization; Feature point extraction algorithm; Feature matching algorithm; UPS; OpenCV

目　　录

引言

随着视觉技术的飞速发展，各种应用场景对于视觉定位技术提出了更高的要求，如无人机、自动驾驶等。视觉定位技术已逐渐应用到智能交通、机器人导航、航空航天、军事等领域。在智能交通领域，视觉定位技术可识别交通信号和驾驶员行为，实现自动驾驶，提高交通安全，减少事故发生。在机器人导航领域，视觉定位技术可帮助机器人识别周围环境，提供导航和路径规划信息。在航空航天领域，视觉定位技术可识别空中目标的位置、速度和姿态，帮助飞行员进行精确导航。在军事领域，视觉定位技术可以辅助战场目标的识别和跟踪，提高军事打击的精确度和战斗力[1]。

随着人工智能的发展，智能交通、机器人导航等领域需要更为精确的视觉定位系统来实现智能交通和精确导航。基于图像特征点的视觉定位系统是目前应用较广泛和高效的技术，解决了因图像质量低而导致的定位不精准的问题，也实现了更快速的视觉定位。因此，视觉定位技术在各个领域的研究和应用，都具有非常重要的现实意义和应用价值。

本文旨在深入探讨图像处理方法、图像特征点提取算法以及图像特征点匹配算法，通过获取到基于树莓派 4b 所组成的图传服务器采集到的图像数据，采用多种传统图像提取算法进行比较，包括 SIFT(Scale-Invariant Feature Transform, 尺度不变特征变换)、SURF(Speeded-Up Robust Features, 加速稳健特征)、ORB(Oriented FAST and Rotated BRIEF)和 AKAZE(Accelerated KAZE)算法，再通过 FLANN(Fast Approximate Nearest Neighbor Search Library,快速最近邻逼近搜索函数库)算法实现图像特征点的匹配，应用尺度空间扩展和 RANSAC 算法来提高匹配准确性，根据提取到的关键点的坐标计算出单应性矩阵，从而得到全局图像中局部图像相匹配的四个角的坐标。通过角坐标计算变换矩阵，实现水平偏移角、垂直偏移角以及直线距离的计算。本文所设计的基于图像特征点的视觉定位系统整体框架如图 1 所示。

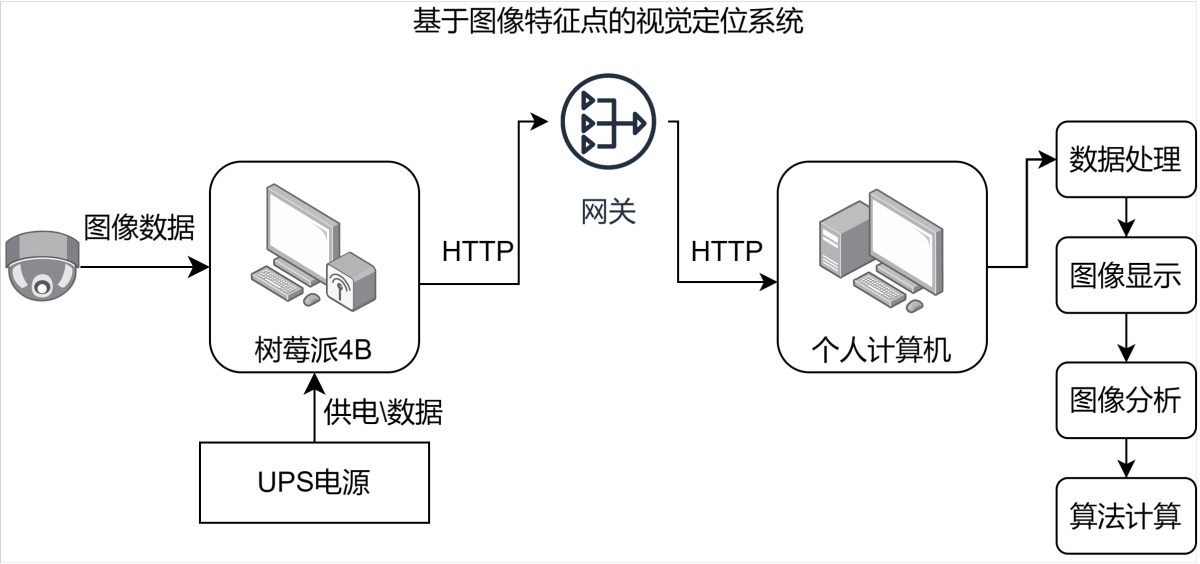


图 1 系统框架

1 文献综述

1.1 国内研究现状

在 2004 年，隋婧和金伟提出并研究了双目视觉技术，分为图像获取、摄像机标定、特征提取、立体匹配和三维重建等步骤，同时展望了其发展[2]。在 2007 年，王国美和陈孝威研究了 SIFT 特征匹配算法，并表明 SIFT 算法具有强大的匹配能力和鲁棒性，是一种良好的特征匹配算法[3]。之后，张锐娟和其团队于 2009 年提出了一种基于 SURF 的图像配准方法，与传统的 SIFT 算法相比，更加高效[4]。2010 年，冯嘉改进了现有的 SIFT 算法，通过简化 SIFT 算子降低了原算法特征点描述符的维度，同时提高了算法的稳定性和匹配度[5]。在 2013 年，肖飞总结了前人的研究成果，深入研究了各种特征提取及特征点描述算法，并提出了一种新的基于图像匹配技术的实时电子稳像算法[6]。最后在 2014 年，刘云鹏和其团队对不同角度的现有图像特征匹配算法，例如 SIFT、SURF、BRISK、ORB 和 FREAK 进行了比较，得出 SURF 和 FREAK 算法表现出更为全面的鲁棒性的结论[7]。

1.2 国外研究现状

图像局部特征研究的历史可以追溯到 20 世纪 70 年代末，当时 Moravec 首次提出了角点特征，采用“兴趣点”概念进行实现[8]。然而，Moravec 角点检测算法存在旋转不变性差和对噪声敏感等问题。为了解决这些问题，Harris 在 1988 年提出了一种新的角点特征算法，该算法具有更高的检测率和重复率，并且对旋转和灰度变化具有不变性[9]。进一步发展局部特征，D.G.Lowe 于 2004 年提出了具有良好的图像旋转、尺度变换、仿射变换和视角变化条件下的不变性优势的 SIFT 局部特征[10]。2006 年，Bay 等人提出了 SURF，通过结合积分图像与 Harr 小波，有效地提高了特征提取的速度[11]。同年，Edward 和 Tom 提出了 FAST 特征提取方法，该方法在仅需 2.3 微秒的时间内实现了特征点匹配。2010 年，Michael、Lepetit 和 Pascal 提出了 BRIEF，该方法通过使用特征点附近随机选择点对的灰度值大小组合成二进制字符串作为特征点描述符[12]。继而在 2011 年，Ethan 等人提出了 ORB，它结合了 FAST 和 BRIEF 方法，并在保持良好性能的同时提高了特征提取效率[13]。

2 相关技术与方法

2.1 图像特征点提取算法

2.1.1 图像特征点提取概述

图像特征点提取算法是一种在数字图像中提取出具有良好辨识性和鲁棒性的稳定图像特征的算法。该算法能够在图像中自动检测出一些具有独特性质的关键点，例如边角点、斑点等，同时给定每个特征点的描述子，可用于匹配和定位。在计算机视觉中，图像特征点提取算法是一种非常重要的技术，在各种视觉应用领域都有广泛的应用，例如目标跟踪、物体识别、3D 重建以及机器人视觉中的定位等各种领域。

在基于图像特征点的视觉定位系统中，图像特征点提取算法扮演着非常重要的角色。首先，利用图像特征点的描述子，可以在不同场景的图像中进行匹配，实现相机姿态的估计和定位。在这个过程中，由于图像特征点的良好的辨识性和鲁棒性，使得匹配和定位的结果更加准确和可靠。其次，基于图像特征点的视觉定位系统中的算法主要分为两种类型：特征点的跟踪和特征点的匹配。在图像中特征点跟踪方面，需要使用特征点的检测算法，例如 Shi-Tomasi 算法、Harris 角点算法、SIFT 算法、SURF 算法等；在匹配方面，需要使用特征点的描述子，例如 SIFT 描述子、SURF 描述子等。因此，在基于图像特征点的视觉定位系统中，图像特征点提取算法是实现相机姿态估计和定位的关键技术之一。本文所设计的图像定位系统即应用到了传统的图像特征点提取算法 SIFT 算法、SURF 算法、ORB 算法、AKAZE 算法，还有基于 Convolutional Neural Networks(卷积神经网络)的 SuperPoint 算法，应用多种图像特征提取算法能够适用多种复杂背景、不同光照度以及快速运动的场景。综上所述，本文所设计的定位系统集成多种特征提取算法是为了兼顾不同场景下的匹配需求，并在实际应用中选择最适合的算法，提高匹配准确率和可靠性，以满足题设中对于误差较低的需要。

2.1.2 SIFT 算法

SIFT 算法是由加拿大英属哥伦比亚大学的 David Lowe 教授于 1999 年提出的。该算法主要用于检测和描述感兴趣点。2004 年，SIFT 基于 David Lowe 教授总结的基于不变量的现有特征检测方法出版。SIFT 是一种局部特征提取和描述算法，可以对图像变换（如尺度、平移、旋转、照明和仿射变换）保持不变性。SIFT 核心思想是将图像匹配转换为特征向量之间的匹配。由于其强大的鲁棒性和快速的计算速度，SIFT 被广泛应用于图像匹配领域，已成为终端引导、数据科学和数据分析等研究领域的热点[14]。SIFT 算法主要分为四步构建尺度空间、特征点监测、特征点方向确定、计算特征描述符。

(1)构建尺度空间

SIFT 尺度空间的构造主要包括高斯金字塔的构造和高斯差分（DoG）尺度空间的构造。其中尺度空间理论表明，高斯卷积核是用于尺度转换的唯一线性变换核。在尺度空间中，图像通过高斯卷积核与灰度值函数进行卷积运算来表达，其尺度因子随之变化，这可用于获得多尺度下的尺度空间表达序列。通过从这些序列中提取尺度空间特征，可以使用拉普拉斯高斯（LoG）算子来表达，如式(1):

$$L(x,y,\sigma)=G(x,y,\sigma)\otimes I(x,y) \tag{1}$$

其中$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{\frac{-\omega^2x^2}{2\sigma^2}}$，符号$\alpha$代表尺度因子，⊗代表函数之间的卷积运算。

式(1)可以使用高斯金字塔表示，对原始图像进行多次高斯模糊处理，每次模糊使用不同的高斯核（具有不同的方差 σ ）。DoG 尺度空间是通过计算相邻高斯金字塔层之间的差值来构建的。具体地说，将具有相邻 σ 值的高斯模糊图像相减，从而得到 DoG 图像。在每个八度（octave）内，将相邻的高斯模糊图像两两相减，形成一组 DoG 图像。每个

八度包含多个 DoG 图像，它们之间的尺度空间分辨率相同，但与相邻八度的分辨率有所不同，如图 2 所示。

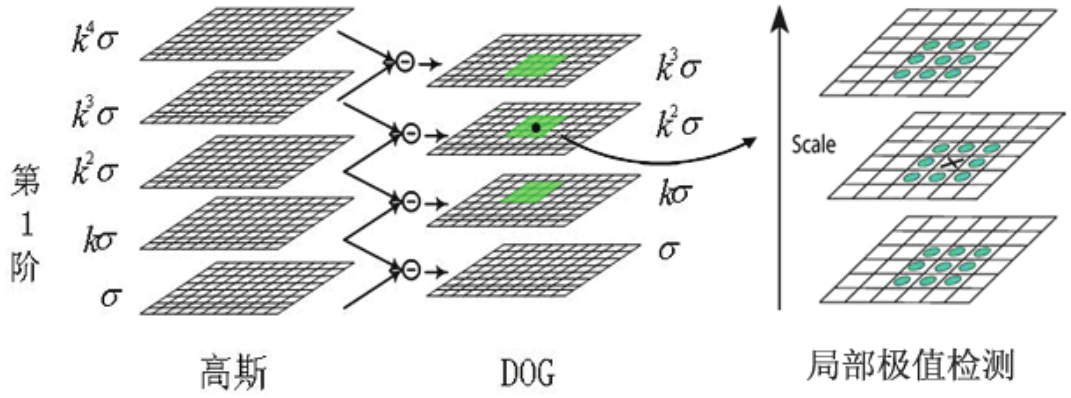


图 2-1 DoG 尺度空间的构造

图 2-1 展示了高斯金字塔作为图像序列结构的一种表示模型。通常情况下，金字塔的组数取值为 4，每组内包含 S+3 张图像子层，其中 S 通常取值为 3。对于金字塔中的第 i 层 (2≤i≤4)，在下一层(i-1)的倒数第二层(s5)进行交错列降采样，获得第一层(s1)。每个金字塔的第一层(s1)通过 Gaussian 卷积核预处理输入图像 $I_{input}(x,y)$，其尺度因子为 σ。每个金字塔的第 j 层(2≤j≤6)是使用尺度因子为 $k_{j-1}\cdot d\sigma$的高斯卷积核对同一金字塔的第 j-1 层进行滤波得到的，其中 k 代表尺度空间的常数因子，$d\sigma$代表基础层的尺度因子。此外，可以尝试在句子中使用代词避免对同一事物进行重复描述。

(2)特征点监测

SIFT 特征点是指 DoG 尺度空间内具有尺度不变性的局部极值点，如图 2-2 所示。对于 SIFT 算法中，中间检测点将与尺度相同且相邻尺度的另外 26 个点在邻域内进行比较，以成功地获得二维图像空间和尺度空间内的极值点，因此通过初始位置检测到的特征点具有位置坐标和尺度坐标。

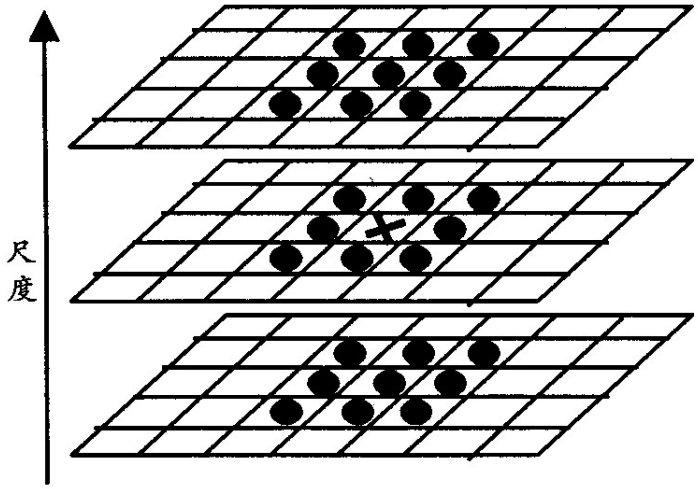


图 2-2 寻找极值点

在极值比较过程中，最顶层和最底层无法进行处理，即无法对边界点进行极值比较。

为了满足尺度变化的连续性，在图像集的边界层上使用高斯模糊方法生成了三幅图像，以完成极值比较。由于变换过程中生成的边缘响应点和噪声点也存在于高斯差分尺度空间中，即图像边缘和噪声点也被识别为特征点。使用类似于 Harris Corner 检测器的方法进行噪声点检测，利用泰勒展开公式及其推导公式，从特征点中删除低对比度和边缘响应点。

(3)特征点方向确定

经过精确定位，特征点被称为关键点。为了通过局部不变特征实现旋转不变性，需要为每个关键点分配一个主方向，该方向仅依赖于图像的局部信息。在此阶段，SIFT 算法在以关键点为中心、以 3 个半径为 1.5 的邻域窗口中进行采样和加权。然后，用模板[0.25，0.5，0.25]连续两次对每三个相邻像素进行加权，并用梯度直方图计算邻域像素的梯度方向，从而可以确定关键点的主方向为相应峰值所对应的方向。如果有其他方向的梯度值比主方向大 80%，则将这些方向确定为关键点的辅助方向，以提高特征匹配的鲁棒性并实现特征描述符的旋转不变性。此方法无疑可以使关键点主方向的分配更加准确，并具有良好的特征描述符辨识度。

(4)计算特征描述符

每个检测到的关键点具有高度的特征描述符差异性，因此可以避免特征向量的突变。此外，特征描述符紧凑而详细，并且对于缩放、旋转和光照等变换具有很强的鲁棒性。此外，通过邻域方向信息的组合，算法的抗噪性能得到了增强，因此能够满足包含位置误差的特征匹配的良好容错性。

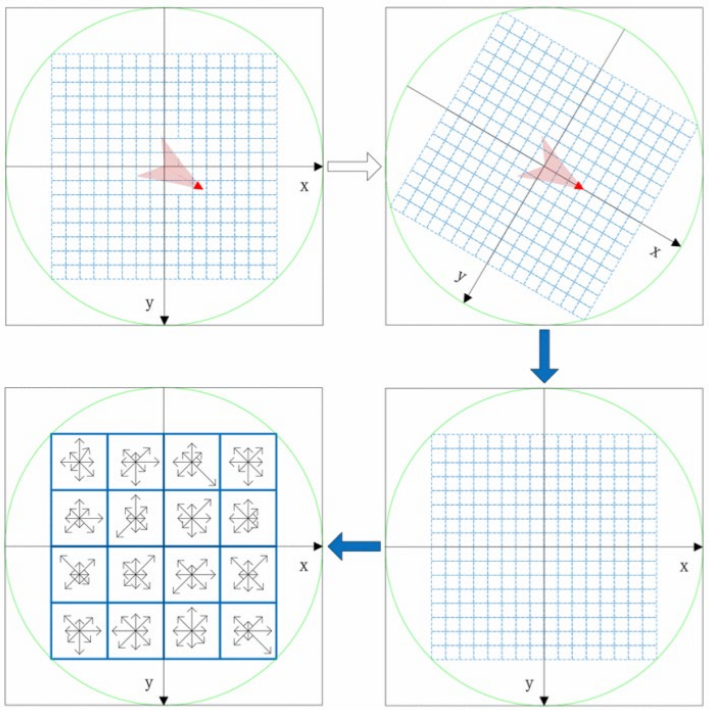


图 2-3 特征描述符的生成过程

如图 2-3 所示，首先将坐标轴旋转到关键点主方向，以确保特征描述符的旋转不变性。然后根据关键点的尺度因子选择相应的 DoG 图像，并将圆形邻域(中心:关键点，半径：$8\sqrt{2}$)分成 44 个子域。随后，采用梯度直方图计算每个子域内 8 个方向(0°、45°、90°、135°、180°、225°、270°和 315°)的梯度模值和方向。

因此，每个关键点可以用包含 4x4x8x128 维特征描述符来表示，并实现尺度不变性。最后，对特征描述符进行归一化处理，以确保对照明变换的强鲁棒性。

2.1.3 SURF 算法

SURF 算法是在 SIFT 算法的基础上进行改进的，SURF 算法与 SIFT 算法相比，具有更快的计算速度和更好的鲁棒。SURF[15]算法的主要实现步骤包括尺度空间的构建、极值点检测、方向分配、特征描述和特征匹配。其中，尺度空间的构建和极值点检测与 SIFT 算法类似，主要区别在于通过 Hessian 矩阵来检测极值点。在方向分配方面，SURF 算法使用了一个旋转不变的算法来确定特征点的方向。特征描述子的计算则采用了 Haar 小波变换，相比 SIFT 算法更快。在特征匹配方面，SURF 算法使用了一种类似 KD Tree 的数据结构，可以实现快速的特征匹配。其与 SIFT 算法的区别有以下方面：

(1)特征检测方式不同：SIFT 算法通过高斯差分来检测图像中的特征点，而 SURF 算法通过 Hessian 矩阵的行列式来检测图像中的极值点。

(2)特征描述子不同：SIFT 算法使用了 DoG 算法来计算特征描述子，描述子维数为 128 位；而 SURF 算法则使用了 Haar 小波变换来计算特征描述子，描述子维数为 64 位，相比 SIFT 算法计算速度更快。

(3)尺度空间构建不同：SIFT 算法使用高斯金字塔构建尺度空间来检测尺度不变性特征点，而 SURF 算法则使用了尺度不变的波特函数来构建尺度空间，增强了算法的尺度不变性。

(4)特征匹配方式不同：SIFT 算法采用 NNDR（Nearest Neighbor Distance Ratio）近邻比值算法进行特征匹配，可以降低误匹配率；而 SURF 算法则采用了一种类似 KD Tree 的算法进行特征匹配，速度更快，匹配精度更高。

但由于 SURF 算法受专利保护，在截止本文截稿前最新的 OpenCV4.7 版本中不存在相关调用的函数，因此本文所设计的定位系统软件所使用的 OpenCV 版本为 3.4。

2.1.4 ORB 算法

针对 SIFT 和 SURF 的实时性缺陷，E Rublee 等人在 ICCV 2011 年提出了 ORB[16]特征检测算法，大大提高了处理速度。ORB 算法基于众所周知的 FAST 算法和 2010 年提出的 BRIEF 特征描述子。其算法的执行步骤如下：首先通过 FAST 算法检测出关键点；然后根据关键点计算每个点的方向和比例尺，使用 pyramid scale space 方法；将图像分成小区域并计算 BRIEF 描述符；最后使用 Harris 算法对关键点进行筛选。相较于 SIFT 和 SURF 算法，ORB 算法的主要区别在于：

(1)速度更快：ORB 算法使用了 FAST 算法进行关键点检测，不需要进行尺度空间的搜索，运算速度更快；

(2)计算量小：ORB 算法采用了 BRIEF 描述符，其计算的是每个关键点周围像素的二进制特征，计算量相对较小；

(3)不受专利限制：SIFT 和 SURF 算法受专利保护，在商业应用中会受到限制，而 ORB 算法不受专利保护，可以随意使用。

总体而言，ORB 算法解决了 SIFT 算法和 SURF 算法的速度慢和受专利保护的问题，而且在一些场景下，特征匹配效果也优于 SIFT 和 SURF 算法,特征匹配效果直接决定最后计算偏移角和直线距离的误差，即在处理某些他图像数据上 ORB 算法的效果比前两者都好。但是，与 SIFT 和 SURF 相比，ORB 算法对于光照变化和视角变化的鲁棒性较差，因此在光照变化明显的条件下，优先选用 SIFT 和 SURF 算法。

2.1.5 AKAZE 算法

SIFT 和 ORB 算法都使用高斯滤波器来构造用于特征检测的线性尺度空间。但是，该方法还可能在过滤噪声的同时使图像模糊并降低图像细节特征的准确性。为了解决这个问题，AKAZE[17]采用了各向异性扩散公式来构造非线性扩散滤波器的尺度空间。该方法使图像的扩散在平滑区域中更快发生，在边缘处更慢发生，从而更有效地保留图像的边缘和细节，其非线性偏微分方程如式 2 表示

$$\frac{dL}{\partial t} = div(c(x,y,t)) \cdot \nabla L \tag{2}$$

其中∇L 表示图像亮度的梯度，div 表示散度，(x,y)表示图像坐标，c(x, y, t)表示传导函数。当 c(x,y,t)恒为 1 时，非线性尺度空间就相当于线性高斯尺度空间，进化时 t 代表尺度参数σ，数值越大则图像表现形式就越简单。

AKAZE 的特征检测步骤与 SIFT 类似，都是先构建非线性尺度空间、再寻找极值点、再计算特征点的主方向、最后计算特征描述符。但不同于 SIFT，AKAZE 算法使用了不同的方法来构建尺度空间，叫做"非线性尺度空间"，这使得它具有更好的适应性和更多的尺度范围。此外，AKAZE 算法还使用了一种叫做"Fast Explicit Diffusion（快速显示扩散）"的技术来进行图像平滑和尺度空间的构建，这也有助于提高算法的运行速度和抗噪性能。另外，AKAZE 算法还可以自适应地选择特征点的大小和方向，进一步提高了其性能。

2.1.6 SuperPoint 算法

SuperPoint[18]算法是由斯坦福大学计算机视觉实验室的 Daniel DeTone, Tomasz Malisiewicz 和 Andrew Rabinovich 在 2018 年提出的。它是一种用于特征检测和描述的神经网络模型，可以快速、准确地检测图像中的关键点并进行描述，被广泛应用于计算机视觉中的 SLAM、三维重建、目标跟踪等领域。

其训练的步骤如图 2-4 所示，在 Daniel DeTone 及其团队的自我监督方法中，预先训练一个初始兴趣点检测器，合成的数据和应用一种新颖的单应适应程序来自动标记来自目标未标记域的图像。生成的标签用于训练一个全卷积网络，该网络联合从图像中提取兴趣点和描述符。

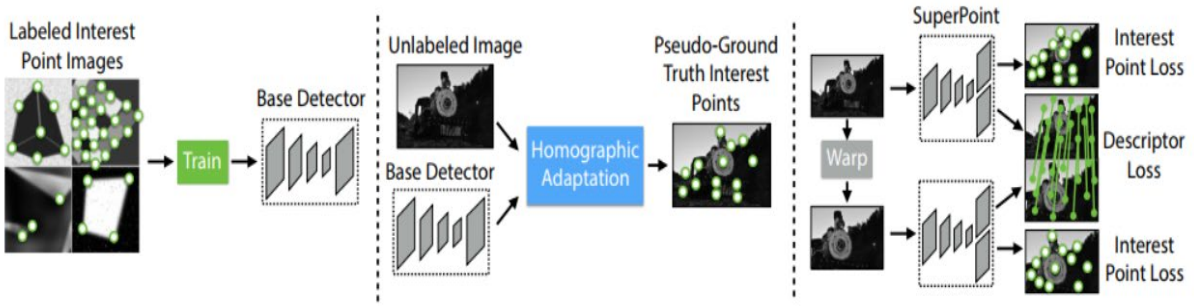


图 2-4 SuperPoint 模型训练过程

结合上述所说的训练过程，Daniel DeTone 及其团队设计了一种名为 SuperPoint 的全卷积神经网络架构，它可以在一个完整尺寸的图像上运行，并在一个前向传递中生成伴随的固定长度描述符的感兴趣点检测结果（请参见图 2-5）。该模型具有一个共享编码器，用于处理和降低输入图像的维度。编码器之后，架构分成两个“头”解码器，分别学习任务特定的权重：一个用于感兴趣点检测，另一个用于感兴趣点描述。大多数网络参数在这两个任务之间共享，这与传统系统不同，传统系统首先检测感兴趣点，然后计算描述符，并且缺乏在两个任务中共享计算和表示的能力。SuperPoint 算法的主要优势是它的速度和准确性。相较于传统的特征检测算法，SuperPoint 算法不需要迭代优化，可以实现端到端的训练，从而大大提高了特征检测的速度。同时，SuperPoint 算法能够在低纹理、光照变化、视角变化等情况下准确地检测和描述特征点，具有很好的鲁棒性。

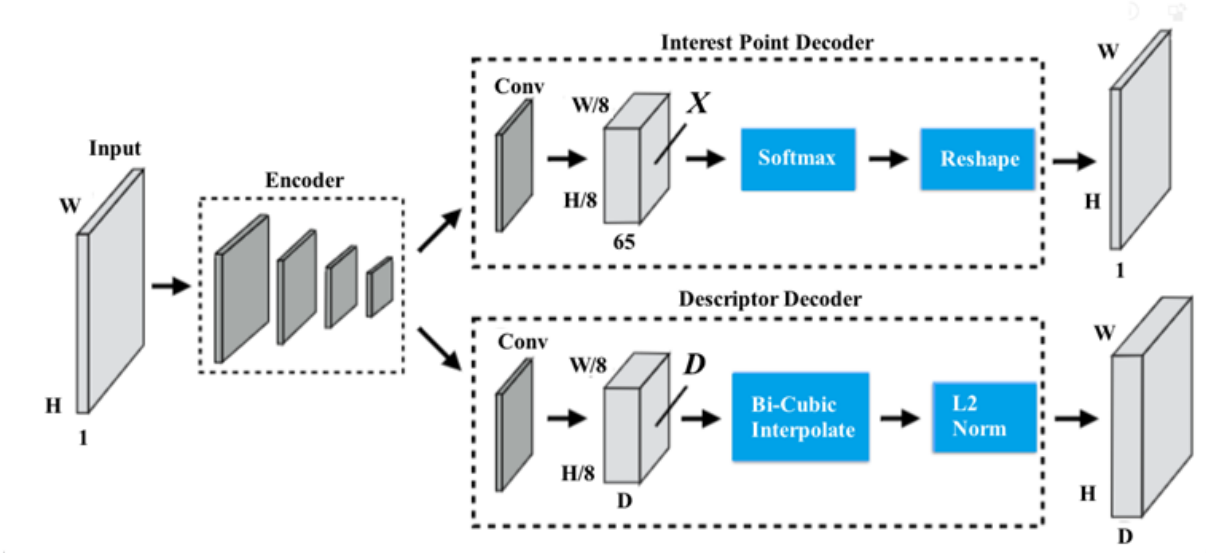


图 2-5 基于 SuperPoint 神经网络架构的兴趣点检测结果

2.2 图像特征匹配算法

2.2.1 图像特征匹配概述

图像特征匹配就是将一张图像中的特征点通过某种方法与另一张图像中的相似的特征点建立对应关系，以便实现图像配准、物体识别、场景重构等计算机视觉任务。在基于图像特征点的视觉定位系统中，特征匹配的作用是通过对两张图像所提取的特征点进行匹配，获得两张图像间的相对位姿或者相对变换关系，进而全局图片与局部图片的偏移角和直线距离。常用的图像特征匹配算法有以下几种：

(1)最近邻匹配算法：该算法将待匹配特征点的特征向量与参考特征点库中的每个特征点进行比较，选择距离最近的点作为匹配点；

(2)基于比值测试的最近邻匹配算法：该算法使用待匹配特征点的特征向量与参考特征点库中前 k 个距离最近的点的距离比值来筛选合适的匹配点；

(3)基于 FLANN 的 Kd-Tree 匹配算法：该算法使用 FLANN 库来进行最近邻搜索，可以快速地计算出待匹配特征点的最近邻点。

(4)基于粗糙匹配的 RANSAC 匹配算法：该算法使用随机选取的一组匹配点，并利用这些点来拟合最优的模型，然后通过计算其他特征点与该模型的拟合误差来筛选出合适的匹配点，并不断迭代优化，得到最终的匹配点。

综合实际情况考量，结合上述所提到的图像特征提取算法，本文所设计的定位系统所使用到的特征匹配算法为 FLANN 算法，将匹配点再通过 RANSAC 算法过滤获取最为准确的匹配点，这种方式极大地保证了后续计算的精确性。同时基于深度学习模型的 SuperPoint 有其配套的特征匹配算法，即 SuperGlue 算法，两则结合能发挥模型的最大效益。

2.2.2 FLANN 与 RANSAC 算法

FLANN 是一种高效快速计算最近邻居(Nearest neighbor)的算法库，k-Tree 是 FLANN 算法库的核心算法之一，它是一种数据结构，用于在高维数据空间中快速搜索最近邻居，尤其适用于大量数据集的最近邻匹配。基于 FLANN 的 Kd-Tree 匹配算法使用 Kd-Tree 作为数据结构，通过利用多个子空间，将数据点分隔在空间中，同时保留空间分割的信息。在进行匹配时，它会首先根据特征向量生成 Kd-Tree，然后根据待匹配特征点在 Kd-Tree 中查找最近邻居点。基于 FLANN 的 Kd-Tree 匹配算法的优点是速度快、可应用于大规模和高维度数据，而且可以根据需要设置不同的搜索参数，但匹配精度并不高，在图像匹配过程中可用于图像特征点的粗匹配。

RANSAC[19]算法是一种模型估计和数据适应性测试的经典算法，由 Fischler 和 Bolles 于 1981 年提出。它主要应用于寻找带有噪声和异常值的数据集中的最优模型参数。在计算机视觉领域中，RANSAC 算法通常被应用于特征点匹配，例如计算 Homography 矩阵来进行图像配准。基于 FLANN 的 Kd-Tree 匹配算法在计算最近邻点时速度非常快，而 RANSAC 算法可以在匹配的过程中去除错误的匹配点，提高匹配的精度。因此，将基于 FLANN 的 Kd-Tree 匹配算法和 RANSAC 算法结合起来，可以得到更

好的匹配结果。具体实现方法是：利用基于 FLANN 的 Kd-Tree 匹配算法得到较多的匹配点，然后根据 RANSAC 算法去除错误的匹配点，最终得到正确的匹配点。

2.2.3 SuperGule 算法

SuperGlue[20]是一种图像特征匹配算法，由 R. Sarlin、D. DeTone、T. Malisiewicz 和 A. Rabinovich 在 2020 年提出。与传统的特征匹配算法不同，SuperGlue 使用可学习的投影模型来估计每个点匹配的概率分布，并利用该概率分布进行最终的匹配决策。它不仅可以匹配 2D 图像，还可以匹配 3D 点云和深度图。SuperGlue 算法的核心是一个神经网络，该神经网络输入两个图像特征点集，输出所有匹配的概率分布。与 SuperPoint 算法一样，SuperGlue 算法也利用深度卷积网络从输入图像中提取特征点。SuperPoint 算法是一种图像特征点检测和描述算法，与 SuperGlue 算法不同，它利用 Harris corner 检测器来检测输入图像中的角点，并利用深度卷积网络从这些角点中提取特征描述子。与传统的特征点描述算法不同，SuperPoint 算法利用可微分双线性池化来提高特征点的稳定性和鲁棒性。SuperGlue 和 SuperPoint 算法可以结合使用， SuperPoint 检测输入图像中的特征点并提取特征描述子，然后 SuperGlue 通过特征描述子在两个图像中进行匹配。这种结合方法在本文所设计的定位系统中发挥了关键性作用，解决了部分场景下(如光照差异大，场景色彩丰富等)不易于推图像特征提取和匹配的问题，使得基于图像特征点的角度、距离计算更为准确，极大地减小了误差，提供了丰富的定位信息。

3 系统设计与实现

3.1 系统的概述

本文介绍图像定位系统是基于图像特征点实现的，分为图像采集和图像处理两个部分。前者通过树莓派 4b、UPS 电源和 USB 双目摄像头实现，后者使用 Windows 操作系统下的自主开发的应用程序实现图像处理，特征点提取、匹配并进行角度和距离计算。

图像采集部分由树莓派 4b、UPS 电路板和 USB 双目摄像头组成。树莓派 4b 采用 ARM Cortex-A72 架构，CPU 主频高达 1.5 GHz，集成了 2GB LPDDR4-3200 SDRAM。由电源管理芯片 BQ25895RTWR 和稳压芯片 TPS61236P 组成的 UPS 部分保证了系统的可靠性。USB 双目摄像头提供了需要处理的原始图像数据。这三部分实现了对实际场景的图像采集。

图像处理部分是一个自主设计的应用程序，运行于 Windows 操作系统。该应用程序通过处理图像和 UPS 电源数据，实现了 UPS 电源数据的显示，图像特征点的提取、匹配，以及角度和距离的计算。应用程序使用 OpenCV 和 Pyside6 库进行开发设计，OpenCV 用于图像处理，提取图像特征点，而 Pyside6 用于实现图形用户界面。综合应用 SIFT、SUFT、ORB，AKAZE 算法以及 FLANN、RANSAC 算法，以及额外采用基于卷积神经网络的 SuperGlue 和 SuperPoints 算法对输入的局部图像数据和全局图像数据

进行特征提取和匹配，通过对算法结果的计算数据综合排序后，得出误差最小以及耗时最小的值。

3.2 图像采集部分

3.2.1 UPS 电路设计

UPS 电源电路设计初衷是为了解决树莓派 4b 本身不带电源而不方便携带的问题，提高系统的可携带性和灵活性。考虑到树莓派 4b 需要大约 3A 的电流，电流规格较大，因此采用了 TI(德州仪器)的电源管理芯片 BQ25895RTWR 和升压芯片 TPS6123，电路原理图及实物图见附录 1。

BQ25895RTWR 是一款适用于单节锂离子电池和锂聚合物电池的高度集成型 5A 开关模式电池充电管理和系统电源路径管理器件。此类器件支持高输入电压快速充电。低阻抗电源路径对开关模式运行效率进行了优化、缩短了电池充电时间并延长了放电阶段的电池使用寿命。而 TPS61236P 是一个高输出可调电压同步升压转换器，能够提供稳定的电压电流输出，以保证树莓派 4b 电源的稳定性。而 UPS 电路板的锂电池容量选择了 6000mAH，经测试，在断开电源的情况下，能允许树莓派 4b 持续运行 3h 左右。通过对电源管理芯片周围电路的电阻电容参数的不断调整，整个 UPS 电源电路具备了以下的优点：

(1)高效的充电效率以及宽电流输入：高效 5A、1.5MHz 开关模式降压充电；2A 充电电流下的充电效率为 93%；3A 充电电流下的充电效率为 91%；针对高电压输入（9V 至 12V）进行了优化。

(2)支持升压操作，以及内置升压转换器：升压模式操作，可调输出电压范围为 4.5V 至 5.5V；具有高达 3.1A 输出和 500KHz 至 1.5MHz 可选频率的升压转换器；5V/1A 输出时的升压效率为 93%。

(3)宽输入电压范围，适配多种电源适配器：支持 3.9V 至 14V 输入电压范围输入电流限制（100mA 至 3.25A，分辨率为 50mA），支持 USB2.0、USB3.0 标准和高电压适配器；通过输入电压限制（最高 14V）实现最大功率跟踪，适用于各类适配器；自动检测 USB SDP、CDP、DCP 以及非标准适配器。

(4)优秀的系统性能：BATFET 控制，支持运输模式、唤醒和完全系统复位；灵活的自主和 I2C 模式，可实现出色的系统性能；高集成度包括所有 MOSFET、电流感测和环路补偿。

(5)高精度和高安全性：集成用于充电模式和升压模式的电池温度检测，支持热调节和热关断。

3.2.2 服务器部署

在树莓派 4b 所部署的服务器方面，本文所设计的 UPS 电源中电源管理芯片 BQ25895RTWR 具备电压电流监测功能，在树莓派 4B 上开发了 Python 代码，通过 I2C 接口实时获取输入输出端电压和锂电池电量情况。此外，还利用 Node-Red 平台建立了

物联网连接，将这些数据实时传输到局域网内的个人计算机中。该平台具有便捷、易扩展以及支持多种协议的特点，用户可通过浏览器访问树莓派 4B 的 IP 地址，直观地观测整个 UPS 系统的运行状态。另外，通过获取 UPD 的 40001 端口发来的数据，用户还可以了解 UPS 状态的详细信息。本文所部署的 Node-Red 数据节点如图 3-1 所示，该节点在 UPS 监控系统中起到了重要的作用，节点包含对电池电压、充电电流、输入电压等数据的处理和显示，同时后台实时计算百分比电量、充电状态等数据。

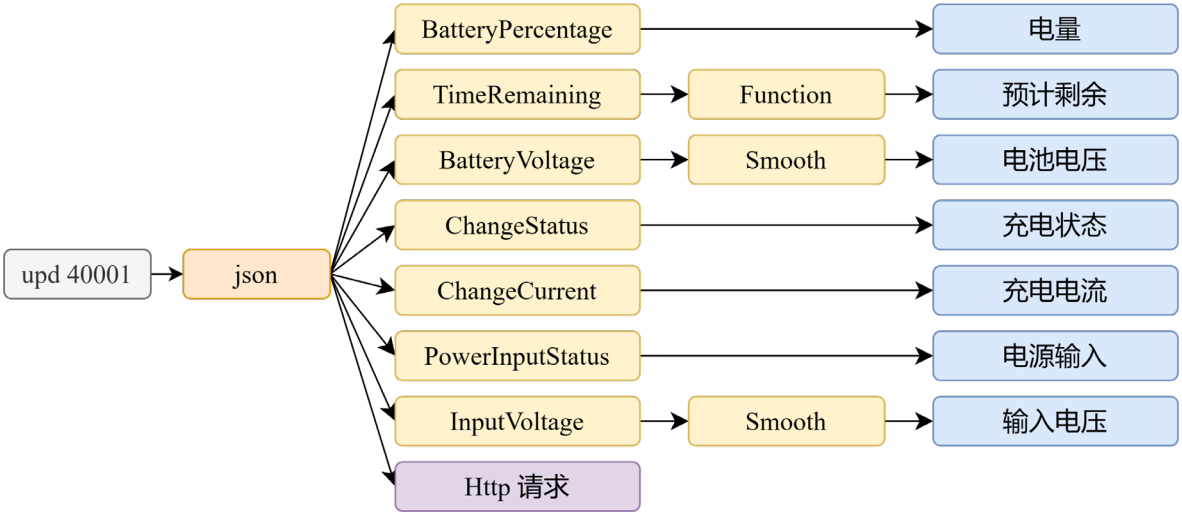


图 3-1 Node-Red 部署的节点框架

在图像传输方面，为了保证图像传输的稳定性和图像质量的可靠性，在图像获取和图像处理部分选择了 OpenCV，在图像传输部分选择了使用 Flask 搭建的 web 框架。其中 OpenCV 是一个开源的计算机视觉库，其内置了许多用于图像处理的函数和算法，如图像识别、目标检测、物体跟踪等，因此可以用它帮助树莓派 4b 高效的处理图像和视频。而 Flask 是一个轻量级的 Web 应用框架，它基于 Python 并使用 Werkzeug 作为 Web 服务器网关接口，Flask 可以帮助我们快速开发 Web 应用程序，易于远程客户端获取到图像数据。将这两者结合起来，我们可以使用 OpenCV 处理图像和视频，并使用 Flask 将处理后的结果在 Web 上展示出来，实现了图像处理和 Web 应用的结合。具体而言，我们可以使用 OpenCV 捕获图像和视频，并使用 Flask 将图像数据传输到 Web 页面展示出来，从而达到了远程图传的效果。

3.3 图像处理部分

3.3.1 方案设计

对于远程客户端软件是一个使用 Python 编程语言编写，基于 OpenCV 和 Pyside6 搭建应用程序，通过 Qt 公司旗下的 Qt Designer 开发工具设计用户基础操作界面并保存为.ui 文件，其软件所使用的图像，字体等资源文件保存在 resources.qrc 文件中，这是一种 Qt 专用的资源引用格式文件，通过 Pyside6-uic 工具将.ui 文件转成.py 文件后，可通过 Pyside6 中内置函数将其解析成软件界面，使得整个程序设计更加的直观。用户的软件设定存储在本地的 Settings.ini 文件中，可通过 Python 程序读取到用户的设定并初识

化程序，使得用户在下次打开程序时不用再次设置。通过 Qt Designer 设计图形界面具备多次开发的优点，并通过后台运行的 Python 程序对图传服务器的图像数据进行获取和处理，实现了软件界面与后台的运行的分离，结构清晰，方便后续的维护，也方便了用户操作。本方案技术的难点在于界面的设计、功能的实现、算法的融合和图像的处理，由于在处理 UPS 系统电源数据时需要不断获取树莓派 4b 的 40001 端口，如果在主线程中运行该部分代码，则会造成线程的堵塞。因此，为不影响软件的正常运行，软件的逻辑设计采用了多线程方式，其中一个线程用于处理 OpenCV+Pyside6 构成的图像处理部分，一个线程用于处理 UPS 数据，整体方案的设计如图 3-2 所示。

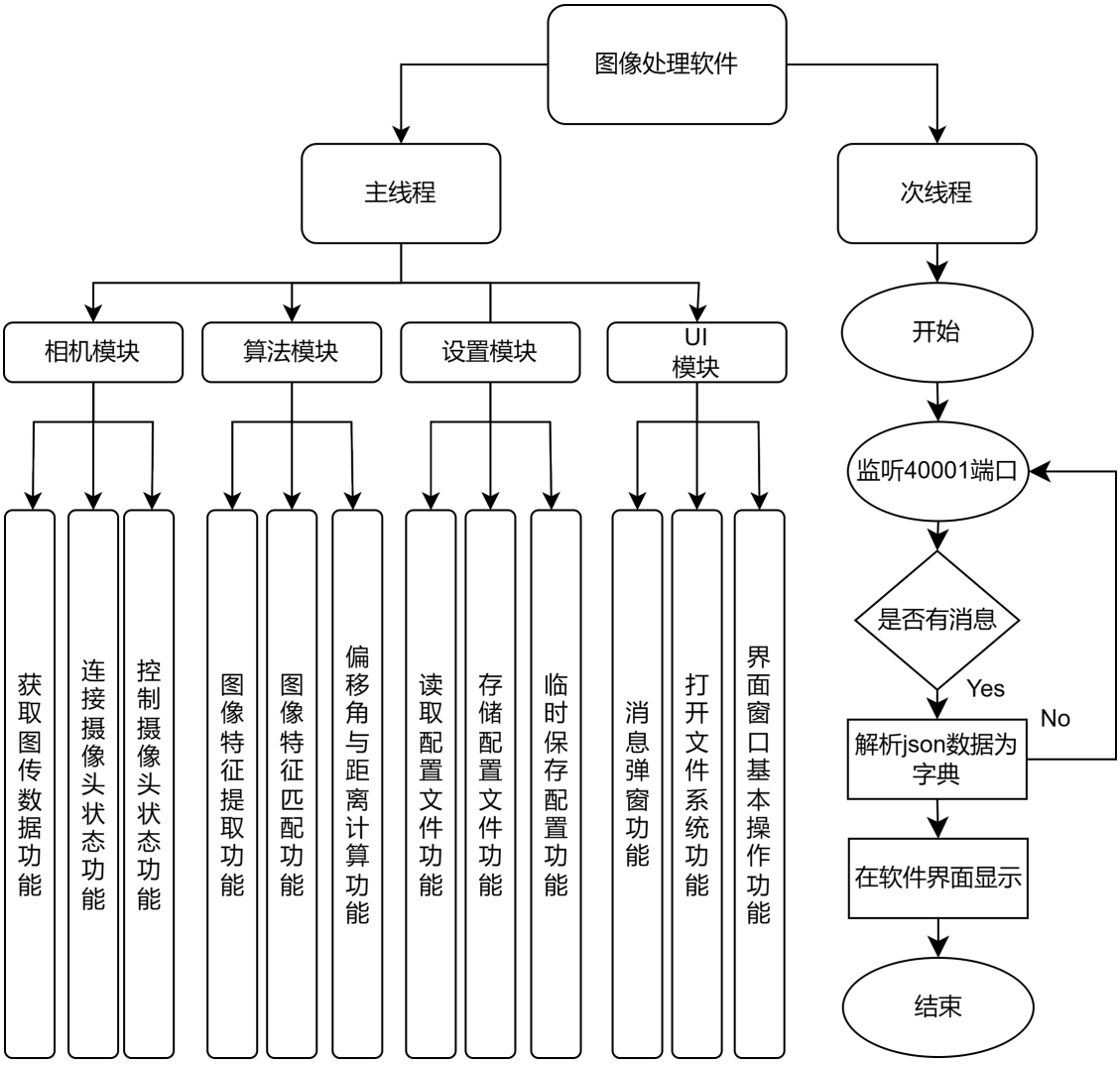


图 3-2 软件总体设计方案

3.3.2 平台搭建

(1)Qt Designer

Qt Designer[21]是Qt公司旗下的一款界面开发工具，通过在Qt Designer窗口中拖拽各种组件，设计窗口布局并结合CSS美化，就可以方便的完成界面设计。使用Pyside6-uic

工具将设计好的.ui软件界面文件转为.py文件，就能完美的与后台的Python代码融合成一个完整的界面，实现软件界面与后台算法运行分离。

(2)OpenCV-Python

OpenCV-Python是OpenCV的Python接口，它是一种基于开源计算机视觉库OpenCV的Python包。它包括许多CV算法与互动开发环境，并且具有简单易用的Python语言接口，可以帮助开发人员更容易地使用OpenCV进行图像处理和计算机视觉操作。OpenCV中包含本文所需的图像特征提取算法SIFT、SURF、ORB和AKAZE算法，以及图像匹配算法FLANN，RANSAC算法等。方便使用Python调用，从而实现图像特征点的提取和匹配，以及计算水平偏移角、垂直偏移角和直线距离。Opncv-Python库可以通过Python中的pip包管理直接下载到，上面所提到的算法库则需要额外下载OpenCV-Python-Contribute库，其中SURF算法由于受专利保护，在OpenCV3.4.1.15之后的版本中不再内置，需要自行编译。因此本文中所使用到的OpenCV-Python版本为3.4.1.15。

(3)Python及Pyside6

Python是一种高级编程语言，具有简洁明了的语法、易读易写、跨平台等特点，适合于快速开发、原型设计、脚本编写等。由于拥有庞大的生态系统和强大的第三方库支持，Python也广泛应用于人工智能、机器学习、数据科学、Web开发、自动化测试、运维管理等领域。

PySide6 是用于基于 Python 和 Qt 技术栈构建跨平台 GUI 应用程序的工具库。PySide6 是 Qt 公司维护的 Qt.py 的分支，支持 Python 3.6+，其提供了 PyQt5 和 pyside6 的功能，可以在不修改代码的情况下轻松地在这两个库中进行切换，并且代码兼容性较好。

PySide6 与 Python 一样，具有高度的可读性、简洁易懂的语法和易于开发的特点，可以为开发人员提供快速、灵活的 GUI 开发工具和库。其主要优点如下：

1、自带 Qt Designer 实现可视化界面，开发 GUI 程序更加快捷、高效。

2、Qt 本身拥有优秀的图形库，可以快速绘制各种图形和界面元素。

3、支持跨平台，应用程序可以在 Windows、Linux、MacOS等多个操作系统中运行。

4、灵活的可扩展性，可以方便地使用 Python 库和 Qt 库中的其他模块以及插件进行扩展。

(4)PyTorch、SuperGlue和SuperPoint

PyTorch 是 Facebook 于 2016 年发布的一个 Python 优先的编程框架，是一个用于强化学习和深度学习的开源机器学习框架。PyTorch 以 Python 语言为基础，具有易于使用、灵活、高度可扩展性等优点，广泛应用于自然语言处理、计算机视觉、深度强化学习等领域。SuperPoint 和 SuperGlue 是计算机视觉中的两个优秀套件，具有卓越的性能和易用性。SuperPoint 是一种基于深度学习的兴趣点检测和描述算法，其具有鲁棒性强、精度高等特点。而 SuperGlue 是一个具有良好鲁棒性和性能的几何匹配库，可以

实现多个图像间兴趣点的匹配，适用于跨时间步的应用程序。SuperPoint 和 SuperGlue 是基于深度学习的算法，因此需要提前安装 PyTorch 以保证 Superglue 和 SuperPoint 算法能够正常的运行。

PyTorch 具有简单易用、灵活性强、可扩展性好等特点，可以帮助研究人员和开发人员更快地开发和实践计算机视觉、自然语言处理以及其他深度学习相关领域。SuperPoint 和 SuperGlue 具有强大的检测和匹配能力，能够有效地定位兴趣点和实现图像之间的匹配。这些模型可以扩展到具有高度重复结构元素的场景，并适用于移动机器人、智能视觉系统等多种领域。本文中应用的 SuperPoint 和 SuperGlue 算法用来适应多种场景下的图像特征点提取和匹配，通过调整关键点提取阈值和匹配阈值来实现高精度的特征点匹配，以保证计算出来的单应性矩阵的准确性，从而减小通过算法计算出来的局部图片相对全局图片的偏移角和直线距离的误差值，实现本文的题设要求。

3.3.3 功能介绍

整个软件界面由Qt Designer进行设计，其后台逻辑由Python+OpenCV进行执行。主要的功能界面分为相机界面、图像处理界面、算法流程演示界面、统计比较界面和用户设置界面，软件功能从获取远程图传服务器接受图像数据和电源数据，到处理图像、计算局部图像和全局图像的偏移角和直线距离，到比较各种图像特征匹配算法的匹配率、耗时和误差率，得出匹配率最高的、耗时最短的以及误差率最小的特征匹配算法，以保证满足本文题目所需实现的内容。

(1)相机界面

相机主界面从左到右的按键功能依次为双目摄像头标定、打开摄像头、保存图像、切换摄像头、实时图像匹配。相机界面的主要功能为接受远程图传服务器发来的图像数据，将其展示在软件界面上，同时用户可自行选定保存图像数据到本地，其实现的流程就是通过OpenCV解析图传服务器的图像帧数据，将其转换二进制图像为可显示的JPG或PNG格式的图像，再由一个定时器定时将OpenCV中获取到的帧图像转换为Format_RGB888格式的QImage对象，再将QImage对象显示在相机界面的一个用于显示图像的QLabel中，因此定时器设定的触发时间决定了图像视频显示的帧数，本文所设计的软件中所设定的触发时间为40ms，即1s时间显示大概25张图片，视频的帧数即为25帧，满足人类眼睛的视觉暂留现象所需符合每秒24帧的标准，因此用户可观察到流程的视频数据。

相机界面还包含了摄像机标定功能和实时摄像机匹配功能，其中摄像机标定功能通过绑定的远程摄像头进行标定，双目立体视觉常常需要进行相机标定以纠正左右相机之间的差异。相机标定是获得相机内外参数（标定矩阵、旋转矩阵、平移矩阵等）的过程，这对于恢复深度信息、进行立体匹配以及3D重建等任务非常重要，双目摄像头标定的流程为接收用户设定的棋盘格规格->获取摄像头图像->对图片进行角点检测，利用角点的二维坐标算出标定矩阵(内参)->根据已知棋盘格的大小和相机光心的坐标，计算外参(旋

转矩阵和平移矩阵)->根据内外参数计算左右相机之间的转换矩阵和投影矩阵->通过计算差异矩阵，对标定结果进行评估。标定的输出结果数据有内参矩阵、外参矩阵、畸变系数、校正矩阵和变换矩阵，其中内参矩阵包含标定的相机光心等内部信息，外参矩阵包含标定的相机相对于固定参考系下的旋转和位移关系，畸变系数即非线性畸变和径畸变参数，校正矩阵和变换矩阵即左右两个相机间的转换矩阵和射影矩阵，可以用于由左视图到右视图的映射。相机标定是一项较为复杂的任务，需要对各类误差进行校正，并且对数据的质量和收集环境要求较高，正确的相机标定可以为后续计算提供精准的数据支持，以满足本文题目中的测量距离低误差要求。

而实时摄像头画面匹配功能是基于SuperGlue和SuperPoint算法运行的一种对网络摄像头图像进行实时特征匹配和运算的技术，由于SuperGlue和SuperPoint是一种基于深度学习的特征匹配算法，并且可以应用显卡的CUDA的高算力进行图像处理和匹配，因此相对于传统的图像特征提取和匹配算法，具有较高的计算精度和速度，可以快速而准确地实现实时匹配，适用于多种视觉任务，相机界面运行的软件界面如图3-3所示。

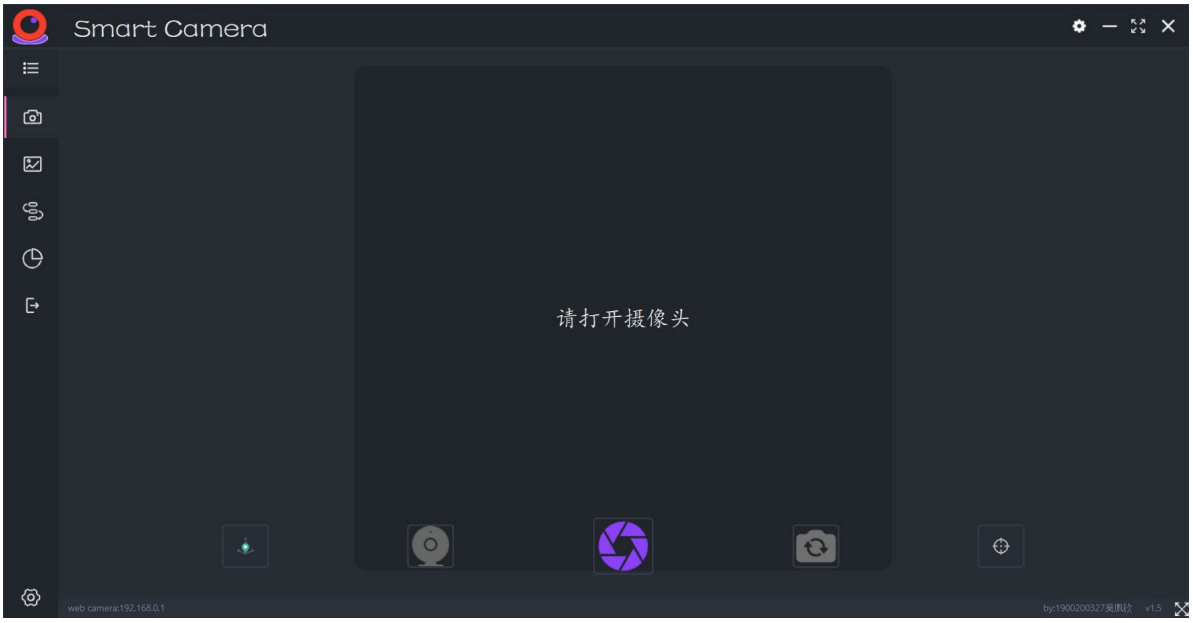


图3-3 软件相机界面

(2)图像处理界面

图像处理界面扮演着相当关键的角色，其中的各项功能直接决定了用户的使用体验。主要功能包括展示用户所指定的需要进行特征匹配的局部图片和全局图片，并依据用户选取的图片进行特征匹配、偏移角度计算、直线距离计算。为了使用户能够轻松使用，该界面还配备了选取局部图片、选取全局图片、特征匹配、偏移角度计算和直线距离计算等按键，使用户可以轻松切换、调整和测试。图像处理界面如图3-4所示，简洁明了的界面设计和智能的操作流程，使得用户可以快速准确地完成各项操作，提高了使用体验和功能效率。

当用户点击"选取图片"的按键时，会打开系统文件选择对话框，用户选择一张需要处理的图片，程序将图片转换为QImage对象，再将QImage显示在窗口的QLabel上，并且将该图像文件的路径存储在软件设置中，用户在下次选择图像时将不再需要选择图像。当用户点击"特征匹配"按钮时，将通过软件设定的特征提取算法进行提取，由FLANN匹配器进行特征点匹配，通过RANSAC算法进行特征点的过滤，以保证匹配的精度，通过单应性矩阵计算匹配角点的坐标，利用OpenCV的polylines函数绘画匹配边框在全局图片中，再将匹配结果通过OpenCV的imshow函数展示出来。当用户点击"偏移角度计算和测量距离"时，会提取上面匹配点的关键点坐标，利用OpenCV中的findHomography函数获取匹配的单应性矩阵，根据变换矩阵获取点坐标，从而计算水平偏移角、垂直偏移角和直线距离，利用OpenCV中的putText函数将结果绘制在结果图片，在由matplotlib的pyplot渲染图片并生成图片显示窗口用于显示结果，当然用户也可以通过设置将结果图片显示在图像处理界面的QLabel标签上。

图3-4 软件图像处理界面

(3)算法流程演示界面

在算法流程界面中，主要功能是将图像特征匹配和计算中的每一步骤进行细化，将每个步骤的结果图片展示在界面上。同时，在用户指定的图像特征提取算法，程序会计算出匹配率、耗时、和计算结果与实际结果之间的误差，为用户提供一个十分方便和直观的交互式用户体验。当用户点击"一键执行"按钮时，程序将根据以下步骤展开：首先，按照图像特征进行FLANN匹配，从而完成特征匹配，然后针对匹配准确度提升考虑，应用RANSAC算法。接着，基于单应性矩阵计算偏移角，最后根据深度图计算水平直线距离。每个步骤都有清晰呈现的效果。总之，本算法流程界面详细的展示了本文所设计的基于图像特征点算法执行的具体流程，方便用户更加直观的感受到算法执行的效果，如果最终效果不理想，也很容易定位到出错的地方。

(4)统计比较界面

在统计比较界面，展示了本文所设计的定位系统软件可使用的5种算法的综合性能表现，包括SIFT算法、SURF算法、ORB算法、AKAZE算法以及SuperGlue和SuperPoint算法，软件呈现的比较界面效果如图3-5所示。定位系统软件集成了多种特征提取和匹配算法的原因是为了极大的可能性来满足题设中所需要的误差阈值。

在图像特征匹配领域中，SIFT算法具有独特的Histogram of Oriented Gradient（HOG）特征描述符和尺度空间极值检测的优势，能够识别出图片的独特特征点，适用于复杂背景下的目标检测和匹配。SURF算法利用高斯滤波和盒子滤波器来分离特征，以加速特征提取速度，同时通过旋转不变性和尺度不变性提高特征的匹配性能，适用于物体外姿态变化较大及反射和阴影的干扰较明显的场景。ORB算法则使用半径为2的BRIEF描述符进行特征提取，并采用FAST点检测算法，具有极高的匹配效率和尺度不变性，适用于系统资源有限、处理高速的场景。AKAZE算法基于非线性尺度空间和加速度计梯度描述符，可以捕捉较小和更活动的特征。相对于SIFT和SURF，AKAZE在模糊、旋转和缩放变换条件下具有更高的鲁棒性和稳定性。SuperGlue和SuperPoint算法则是基于Convolutional Neural Networks的深度学习算法，利用超像素和相似性度量应用于几何匹配，具有较高的鲁棒性、精度和实时性，尤其适用于复杂背景、低照度和快速运动的场景。统计界面的设计是为了展示不同特征匹配算法下的呈现效果，综合匹配率，耗时等多种影响局部图片与全局图片匹配效果的因素，挑选出最适合当前情况下的最优解，并获得最低的误差值。综上所述，本文所设计的定位系统软件集成多种特征提取和匹配算法是为了兼顾不同场景下的匹配需求，并在实际应用中选择最适合的算法，提高匹配准确率和可靠性。

图3-5 软件统计界面

(5)用户设置界面

随着软件界面的迅速更新换代，为符合现代用户的操作习惯，用户的设置界面分为左侧栏和右侧栏，并且带有伸缩功能，其呈现的效果如图3-6所示。其中左侧栏位为主要的软件功能设置，其设置分为图像保存设置、算法参数设置以及额外的设置板块，软件的设置界面如图3-6所示。

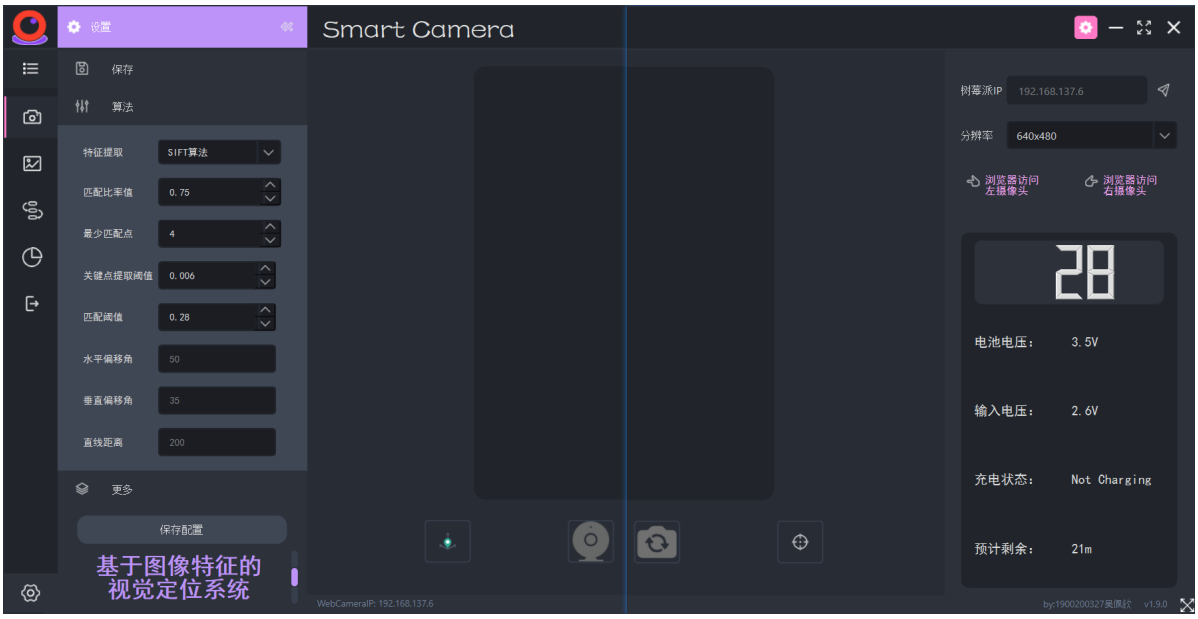


图3-6 软件侧栏设置界面

图像保存设置板块可设置用户保存的图像目录路径，以及图像保存的文件名是否包含后缀名，当用户选择不包含后缀名时，保存的图像固定为globalImage.png，localImag-e.png，分别代表着保存的全局图片和局部图片，当用户保存图像时会自动覆盖前一张图片，而当用户选择包含后缀名时，保存的图像文件名则会以当前的时间撮的后5位数为后缀名，用以区分保存的图片。

而算法参数设置板块可设置算法的一些参数设定，如RANSAC算法计算时的匹配点率(匹配点率越高，匹配点的阈值越高，但也可能会丢失一些正确的匹配点；相反，匹配点率越低，匹配点的阈值越低，一些不匹配的特征点也可能被判定为匹配，从而影响后面计算的精度，本定位系统软件所设定的匹配点率为0.7)、局部图片和全局图片被认定为匹配的最小匹配点数(由于对特征点进行单应性矩阵计算时所需要的最小特征点数量为4，因此本定位系统软件所设定的匹配点阈值为4)、SuperGule和SuperPoint算法中对图像特征点提取的阈值及特征点匹配的阈值(在SuperPoint中，特征点提取的阈值主要是指检测图像中潜在角点的响应分数阈值，一般设置在0.001-0.015之间。该阈值越低，能够检测到越多的角点，但也同时会增加噪声和针对非关键及重复特征的匹配的错误率。SuperPoint的特点在于，它通过使用堆叠卷积操作从底部到顶部的逐层金字塔中提取特征，同时使用非极大值抑制（NMS）进一步过滤掉靠近候选点的响应小于它们的周围点，从而保留最稳定的关键点。在SuperGlue中，特征点匹配的阈值主要是指特征描述符之间

的相似性得分，一般设置在0.1-0.9之间。该阈值越高，需要匹配的特征点对应得越精确，但同时也要求匹配特征点对之间的相似度达到更高的标准。如果该阈值太高，匹配的点对将减少，且精度会降低。与传统方法相比，SuperGlue采用了一种超像素级别的匹配策略，这有助于准确匹配不同位置，大小和姿态的目标，并且通过全局消除错误匹配和选取最佳匹配结果来提高匹配准确率)。算法参数的设置板块还包括用于计算误差的实际图片的水平偏移角、垂直偏移角和水平直线距离。总之，算法参数的设置板块还是十分重要的，它不仅保证了程序能够正常运行，还能通过调整算法执行的参数，来保证匹配的精度和准确性，如通过调整阈值可以控制检测到的特征点数量和噪声水平，从而提高识别的准确性，调整阈值可以控制匹配特征点对的数量和匹配的准确性。

软件设置板块的右侧栏则主要为界面功能设置，包括设置远程图传服务器的IP地址设置图传摄像头的分辨率，最小分辨率为640x480，最大分辨率为1920x1080；还有从浏览器访问摄像头的功能按钮，当图像数据无法正常获取时，可通过浏览器访问到图传服务器所搭建的Web页面是否正常运行，从而排除错误原因。同时服务器所发来的UPS系统数据也将在右侧栏中展现出来，包括的内容有电池电量，输入电压，输出电压，充电状态以及预计剩余时间，使用户能更好的观测到UPS系统板的运行状态，方便用户自行判断是否要给树莓派4b进行充电。

3.3.4 软件工程目录结构

整个软件的工程目录主要分为图像界面代码文件和操作逻辑代码文件。为了方便代码版本的管理，整个项目利用了Git软件进行了仓库初识化，利用Git的一些操作指令即可快速的进行代码的回滚操作或更新迭代，有效避免有些操作导致了软件无法运行的问题。项目中的modules文件夹存放着一些Python类对象模块和UI界面操作的模块，模块化的操作可以很方便的日后添加功能和日常管理。Static文件夹主要存放软件所需的一些图像，字体等静态资源。Images文件夹主要存放用户保存的图像数据，如果该文件夹不存在，软件在初识化的时候回自动进行创建，并指定该目录为图像存放的默认根目录。Main.py文件是整个程序的入口，README.md文件对整个工程进行说明，工程文件夹结构和个文件的介绍如表1所示。

表1 项目文件目录结构说明

名称	类型	介绍
.git	文件夹	记录当前 Git 仓库的版本管理信息
Static	文件夹	存放软件所需的一些图像，字体等静态资源
Image	文件夹	用户保存图像的默认根目录
Left	文件夹	保存左摄像机标定所需的图片
Right	文件夹	保存右摄像头标定所需的图片

续表1 项目文件目录结构说明

名称	类型	介绍
Moduls/SuperGluePretrained Network	文件夹	SuperGlue算法工程目录
Moduls/app_camera.py	文件	相机中调用的算法函数程序
Moduls/app_settings.py	文件	读取和保存用户软件设定程序
Moduls/ui_functions.py	文件	图形界面处理函数程序
Moduls/ui_main.py	文件	由.ui转换来的软件界面文件
Moduls/rasp_ups.py	文件	多线程读取UPS数据程序
Settings.ini	文件	保存用户软件设定配置程序
Main.py	文件	软件运行时的入口程序

4 实验与分析

4.1 实验环境与数据集

对于基于树莓派4b的图像采集部分，其树莓派4b烧录的是官方基于Debian Linux发行版所设计的Raspberry Pi OS（前身为Raspbian），它是专门为树莓派定制的操作系统，目前最新版本是Raspberry Pi OS是基于Debian 11 "Bullseye"操作系统构建。Raspberry Pi OS支持ARMv7和ARMv8架构的树莓派，其中树莓派4B使用的是ARMv8架构。树莓派4b所接入的网络是校园网，校园网的信号遍布整个校园，并且通过校园网的管理后台可以很方便的观察到树莓派4b的IP地址，方便后续的程序调试。通过shell软件接入树莓派，将ups.py以及Node-Red平台的节点Json数据部署在树莓派4b上，使用python运行代码使得树莓派的flask服务器得以运行。

而对于图像处理部分,软件运行在系统为windows11，CUP为Intel(R) Core(TM) i7-9750H，GPU为RTX1650的个人计算机上。这颗i7-9750H是英特尔发布的一款高性能移动处理器，采用了基于14纳米工艺的Coffee Lake架构，拥有6个物理内核和12个线程，基础时钟频率为2.6 GHz，最高可达4.5 GHz。此外，i7-9750H还支持英特尔的睿频加速技术，可以根据需要提高CPU的主频，以提高单线程性能和多线程效率。在GPU方面，i7-9750H采用了英特尔UHD Graphics 630集成显卡，它支持DirectX 12和OpenGL 4.5等主流图形API。此外，它还支持Intel Quick Sync Video硬件加速解码和编码，可以大幅提高视频编辑、转码和流媒体应用的处理速度。拥有较高的核心频率和强大的多线程处理能力，可满足高性能计算和图形处理的应用需求。而RTX1650显卡是英伟达推出的一款中端显卡，基于图灵架构。它采用12nm FinFET工艺，拥有896个CUDA核心，56个纹理单元和32个光栅单元。RTX1650显卡的核心频率为1485MHz，加速频率为1725MHz。它配

备了4GB GDDR6显存，频率为12 Gbps，256位宽度，带宽达192 GB/s。相较于上一代显卡，RTX1650支持英伟达的最新图形技术，如光线追踪、DLSS等，可以提供更精致的图形效果。此外，它还采用了NVIDIA Turing架构的Tensor核心和RT核心，可以提供更强大的AI和光线追踪性能，支持DLSS技术可以在不降低图形质量的情况下提高游戏性能。总的来说，RTX1650有着较好的算力支持，对于需要运行卷积神经网络的SuperGlue和SuperPoint算法来说有着不错的技术支持。

而数据集部分，总分为3种图像集。一种图像集是OpenCV示例项目中所提供的各种参考图像，这类图像集的特点是图像特征明显，并且图像已经经过OpenCV官方进行灰度锐化处理，算法的匹配效果更加明显，避免了实际场景的过多干扰。第二种图像集是基于C4D建模软件对模型所渲染的图片，通过调整C4D中的摄像机角度，以获取不同角度下的局部图片，这种情况下局部图片相对与全局图片的定位信息最为精准，即能够准确知道偏移角度和水平直线距离，这种条件下能够精确的计算到通过算法计算的偏移角度和直线距离相对与实际的误差值。最后一种图像集则是基于树莓派4b所搭建的web网络摄像头所采集的图像，这种图像受摄像头参数，网络情况以及场景的光照条件所影响，所受到干扰因素最多，但也能最大限度的展示了算法在实际情况下的匹配情况，以及在干扰因素较多的条件下计算误差。

4.2 实验结果及分析

4.2.1 UPS数据可视化

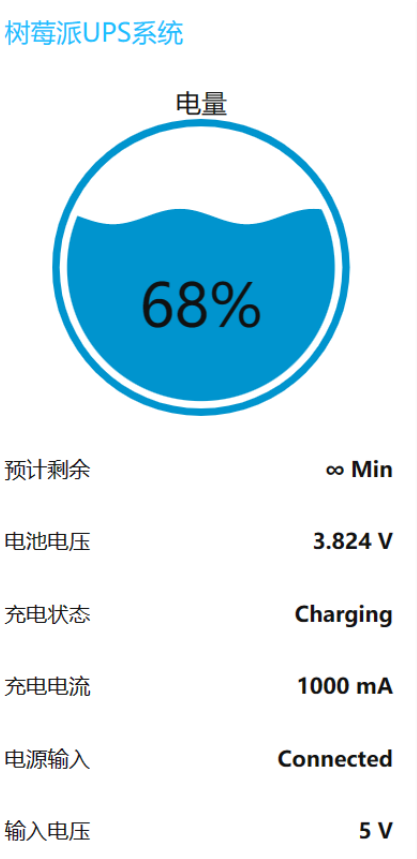


图4-1 UPS数据可视化图

运行树莓派4b上的Flask Web服务器以及Node-Red后，通过校园网的管理后台知道了树莓派4b的IP地址，通过运行windows系统的软件，软件后台通过监听40001端口获取数据，数据包含UPS电源系统的信息及树莓派的IP地址，并将信息打印在控制台，通过该IP地址软件可以自动连接Web摄像头。通过该IP地址可以通过浏览器的访问URL(http://{树莓派IP}:1880/ui)从而访问到UPS电源的网页数据。并且通过软件右侧栏的从浏览器打开摄像头按钮即可快捷打开浏览器访问到web摄像头传过来的图像数据，其传过来的电源数据经过浏览器渲染后可直接浏览，其结果如图4-1所示。

4.2.2 图像处理软件

(1)相机界面功能情况

运行在windows系统的软件工程通过后台运行main.py文件即可将软件运行起来，软件在初次打开时会读取Settings.ini用户配置文件，将配置设定在软件中生效，并且会定时监听40001是否有数据传来，当有IP数据传来时，会自动连接树莓派4b的web摄像头，通过相机界面的打开摄像头按钮即可将画面显示在窗口中，如图4-2所示。

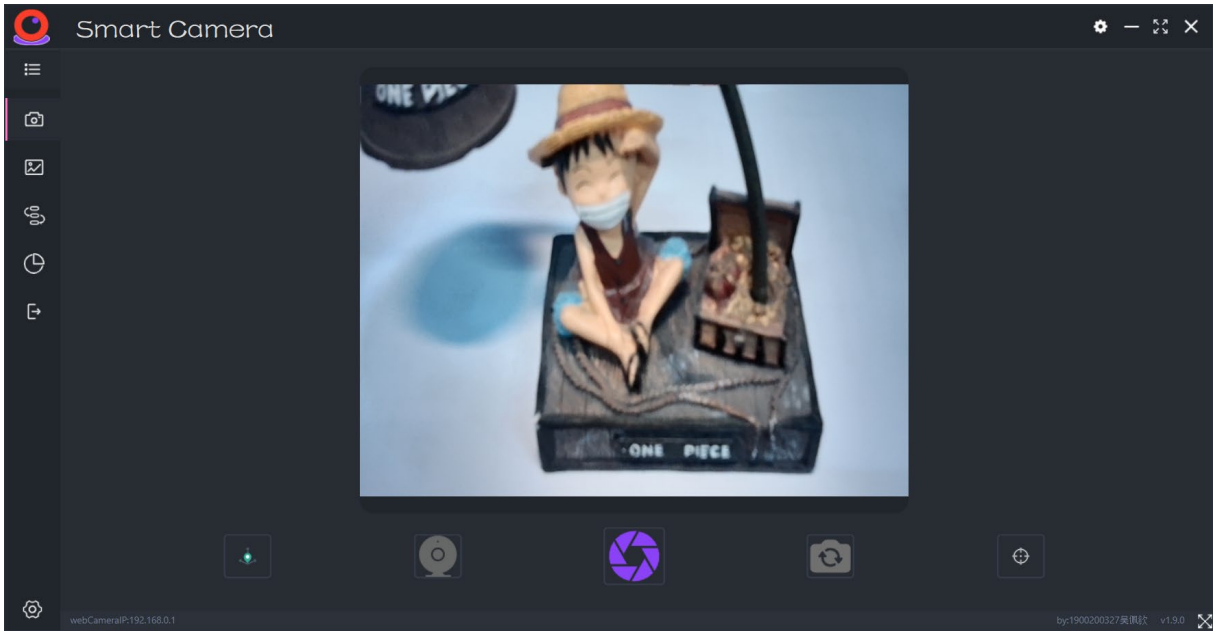


图4-2 软件正常接收到图像数据

并且该界面的保存图像，切换摄像头按键功能也能正常运行。点击标定按键后，弹出输入框，输入所标定棋盘的规格以及每个黑格的实际尺寸后即进入双目摄像头的标定界面，软件会新建一个1280x480的窗口，用于并排显示双目摄像头的图像画面，将黑白格棋盘展示在摄像头面前时，程序会自动定位黑格角点坐标，并计算图像的棋盘规格是否为所需标定的棋盘规格，当符合时会将当前图片帧分别保存工程的left、right文件夹中，用于后面计算双目摄像头的畸变参数，此时参考图数量加1。通过不同角度展示棋盘，能增加畸变参数的准确性，当参考图数量达到100张时，程序结束图片获取并进入标定参数的计算，最终的计算结果将存储在calibration_params.yml文件中，用于后

续对摄像头拍取图像进行畸变校正，经过校正的图像能最大限度的减小相机对图像质量的影响，最大的保证后续计算直线距离的准确性，软件标定过程如图4-3所示。

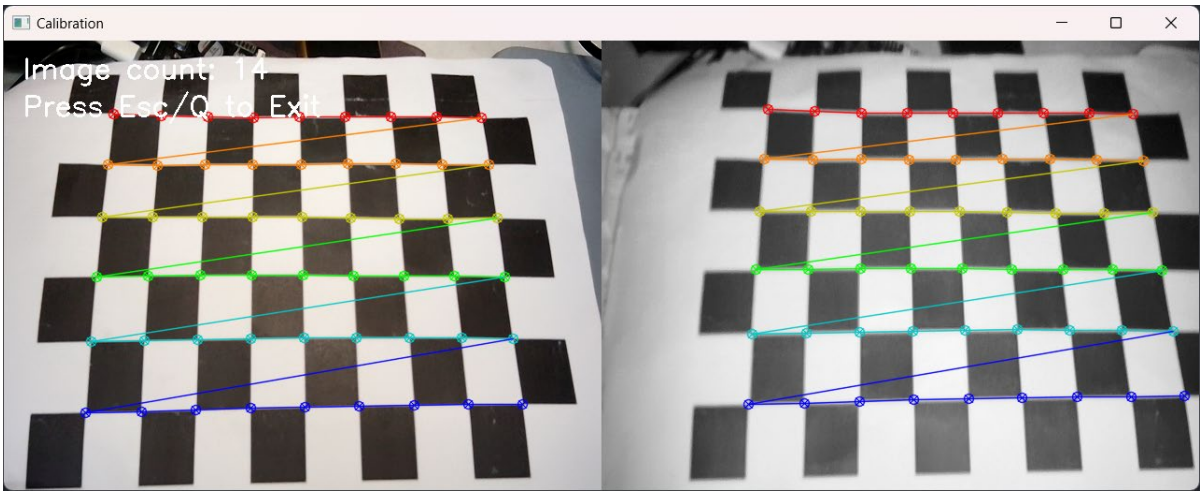


图4-3 相机标定窗口

通过上面的测试说明树莓派4b所构建的图传服务器运行正常，并且软件能正确获取到图像数据。

(2)图像处理界面

图像处理界面主要是读取保存的图片或是需要进行特征定位的图片，然后对图片进行灰度处理，在通过特征提取算法及特征匹配算法对局部图片和全局图片特征点进行提取和匹配，好的匹配点将会用于单应性矩阵的计算，再通过单应性矩阵计算局部图片在全局图片中匹配的相应角点坐标，最后通过相机标定所得的畸变参数计算深度图，从而获取局部图片与全局图片之间的水平直线距离，其效果如图4-4所示(匹配效果使用的是SIFT算法，其余算法的效果在后面的统计界面中展示)。

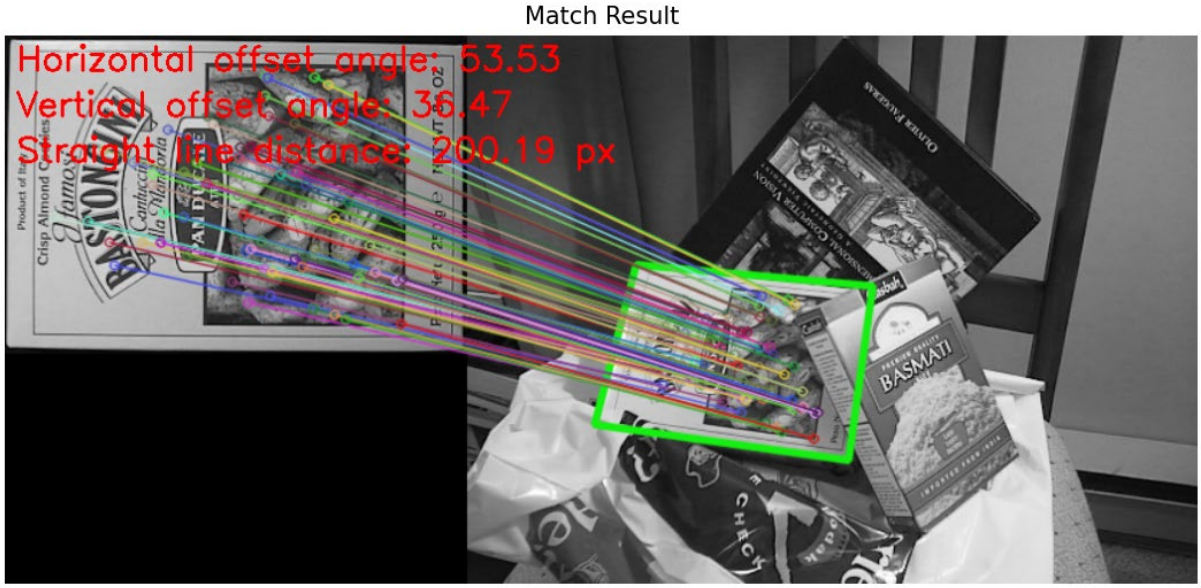


图4-4 基于SIFT算法的图像处理结果

(3)算法流程演示界面

在该界面中，算法程序运行每一步效果以及最终呈现的结果都能够直观的浏览到。每种算法的运行步骤大致分为5步：特征提取->FLANN特征匹配->RANSAC过滤->根据单应性矩阵计算偏移角->根据深度图计算水平直线距离。通过界面上的一键执行按钮，程序自动运行，每完成一步，程序会将结果显示在窗口中，并堵塞主线程继续运行，只有当用户关闭窗口时程序才会继续运行。这样即展示了程序能够正常运行，也直观的展示了算法在图像特征提取和匹配中的作用及效果。当算法最终运行效果不理想，用户也能在后续能够及时的发现算法在哪一步执行的不理想，并及时的做出调整，如修改特征提取算法、提高特征匹配阈值等。算法程序每一步的效果图详见附录2。当算法运行结束时，程序也会将算法的特征点匹配率、耗时以及误差值显示在窗口的右边，方便用户了解各种信息，同时也可以通过该界面中的按钮设置实际的偏移角和直线距离用来计算误差。计算误差的公式如式3所示。

$$\frac{c}{r \times 180^{\circ}} \times 100 \tag{3}$$

式3中c表示通过算法计算的偏移角度，r表示实际的偏移角度，当通过式子计算出来的值小于5%时，则说明算法的匹配效果良好，且满足本文题设中的要求。如果计算出来的值大于5%时，则说明匹配效果不佳，需要根据实际情况对算法的参数进行调整。

(4)综合统计界面

软件统计界面主要是对现有的算法进行一个综合的统计排序，挑选出现有的最佳的匹配结果，从匹配率、误差值、耗时情况等因素综合考虑，获取满足题设的最佳算法。在以OpenCV中的一个示例图片为参考，其每个算法的匹配效果及软件对算法的排序情况如图4-5和表2所示，并且最终的结果图见附录3。

图4-5 软件对各算法结果统计图

表2 各算法计算结果表

算法	SIFT	SURF	ORB	AKAZE	SuperGlue
匹配点/特征点	79/603	42/786	35/453	28/383	24/307
耗时(s)	0.09	0.45	0.04	0.4	0.38
水平偏移角(°)	53.55	53.96	53.78	53.72	53.50
水平误差	0.86%	1.09%	0.99%	0.95%	0.83%
垂直偏移角(°)	36.45	36.04	36.22	36.28	36.00
垂直误差	1.92%	1.69%	1.79%	1.82%	1.67%
直线距离(px)	200.04	199.14	198.00	198.03	199.00
距离误差	0.58%	0.80%	2.56%	1.69%	1.24%

5 结论与展望

5.1 结论

经过对定位系统的结果分析表明，基于树莓派4b的图传服务器能够正常传输图像数据，并且由UPS系统进行不间断供电，增加了树莓派4b的可便携性。同时基于OpenCV和Pyside6所设计的软件也能在windows系统上正常运行，并且通过算法正常获取局部图片和全局图片的特征点并进行匹配，后续对特征点进行偏移角和直线距离进行计算，误差均在5%以内。通过对上述表2所示的统计结果进行分析可得出以下结论：

(1)对于匹配点/特征点：从匹配点和特征点的数量来看，SuperGlue算法在匹配点数量上明显优于其他算法，远高于SIFT、SURF、ORB和AKAZE算法。这说明SuperGlue算法在特征点提取和匹配方面具有较高的性能。

(2)对于耗时：在耗时方面，ORB算法表现最佳，仅需0.05秒，而SuperGlue算法耗时最长，达到0.75秒。这表明在实时性要求较高的应用场景中，ORB算法可能更适用。

(3)对于水平偏移角及误差：从水平偏移角和水平误差来看，SuperGlue算法具有最小的误差，仅为0.23%。其他算法的误差相对较大，但整体上也控制在较低水平。

(4)对于垂直偏移角及误差：在垂直偏移角和垂直误差方面，SuperGlue算法表现较好，垂直误差为1.75%，仅次于SURF算法的1.63%。其他算法的误差相差不大，都在2%以内。

(5)对于水平直线距离：关于直线距离和距离误差，SuperGlue算法的误差为1.24%，相对较小。其他算法的误差也在可接受范围内，ORB算法的误差最大，达到2.56%。

综合分析，SuperGlue算法在匹配点数量、水平偏移角误差和垂直偏移角误差方面均表现出较好的性能。但在耗时方面，其表现较差。因此，在对实时性要求不高的应用场景中，SuperGlue算法具有较高的精度和稳定性。而在实时性要求较高的场景下，可以考虑使用ORB算法。总体来说，不同算法在特定场景下具有各自的优势，可以根据实际需求选择合适的算法进行视觉定位。

5.2 展望与未来工作

在本文中，我成功地实现了一种基于图像特征点的视觉定位系统，通过对多种特征点提取和匹配算法的研究和实验，为实际应用场景提供了有力的技术支持。然而，随着科技的不断发展和实际应用需求的多样化，我认为该视觉定位系统在未来还存在以下几个方面的优化和拓展空间：

(1)算法优化与融合

虽然本文对多种特征点提取和匹配算法进行了研究和实验，但仍有优化的空间。在未来工作中，可以尝试对现有算法进行改进，提高其性能，或者研究融合多种算法的方法，充分利用各算法的优势，实现更高精度和实时性的视觉定位。

(2)深度学习方法的应用

随着深度学习技术的快速发展，基于神经网络的特征点提取和匹配方法在视觉定位领域取得了显著的成果。在未来工作中，可以尝试将深度学习方法应用于视觉定位系统，进一步提高定位精度和鲁棒性。

(3)多传感器融合

为了提高视觉定位系统的性能和稳定性，可以考虑引入其他传感器，如惯性测量单元（IMU）、GPS等。通过多传感器数据的融合，可以在视觉信息受限时提高定位系统的稳定性和可靠性。

(4)动态环境下的视觉定位

本文的视觉定位系统主要针对静态环境进行研究。然而，在实际应用中，定位系统可能面临动态环境的挑战，如移动物体、光照变化等。因此，在未来工作中，可以研究动态环境下的视觉定位方法，提高系统在复杂环境中的适应性。

(5)大规模场景下的视觉定位

当前的视觉定位系统主要针对小规模场景进行研究。在未来工作中，可以探讨将视觉定位技术应用于大规模场景，如城市导航、室外环境等。这需要研究高效的地图表示、地图更新和数据管理方法，以应对大规模场景带来的挑战。

总之，本文实现的基于图像特征点的视觉定位系统为实际应用提供了有力的技术支持。在未来工作中，我们将继续优化和拓展视觉定位系统的性能，以满足不断变化的实际需求。通过对算法的优化与融合、深度学习方法的应用、多传感器融合、动态环境下的视觉定位以及大规模场景下的视觉定位等方面的研究和实践，我们期望视觉定位技术能够在智能交通、无人机、机器人导航等领域发挥更大的作用，为人们的生活带来更多的便利和价值。

谢 辞

在本次论文的写作过程中，我深感自己所受到的关爱和支持无以言表。在此，我想向所有关心、帮助过我的人表示衷心的感谢。

首先，我要向我的导师表示最真挚的感激之情。在论文的选题、设计和撰写过程中，李教授给予了我无私的关怀和悉心的指导。每当我遇到困难和问题时，他总是耐心地为我解答，让我不断地克服困难，取得进展。在本科阶段，正是在李教授的指导下，我学到了许多宝贵的知识和方法，更懂得了如何独立思考和解决问题，这些都将成为我今后学术生涯中的宝贵财富。

同时，我也要感谢我的学长们，他们在论文的设计和实验过程中，给予了我很大的帮助。在我遇到技术难题时，他们总是不吝分享自己的经验和见解，帮助我找到问题的解决方案。有时甚至在深夜，他们也愿意为我答疑解惑，让我深感温暖。在他们的帮助下，我成功地解决了许多在设计时遇到的问题，使得论文的研究得以顺利进行。

此外，我还要感谢我的同学们和朋友们，他们陪伴我度过了这段美好的大学时光，给予我关爱和支持。在我忙碌于论文时，他们总是耐心地倾听我的倾诉，提供宝贵的建议。他们的关心和鼓励成为我在本科阶段学业上取得成功的重要动力。

最后，我要感谢我的家人，他们一直以来都是我人生中最坚实的后盾。在我遇到困难时，他们总是给予我关爱和支持，让我勇敢地面对挑战。他们的鼓励与信任，使我在本科阶段取得了诸多成绩，为未来的学术生涯奠定了基础。

请原谅我在此无法一一列举所有帮助过我的人，但请相信，我将把对你们的感激之情永远铭记在心。在未来的道路上，我将继续努力，用所学到的知识和技能去回报关心和帮助过我的每一个人。

在接下来的研究生阶段，我将继续以谦逊的心态去学习，不断提高自己，努力在科研道路上取得更高的成就。我深知在未来的学术探索中还有许多未知的挑战等待着我，但有了这段本科阶段的经历，我相信自己已经具备了应对未来挑战的勇气和智慧。

参考文献

[1] 孟晓桥，胡占义. 摄像机自标定方法的研究与进展[J]. 自动化学报, 2003, (01): 110-124.

[2] 隋婧，金伟其. 双目立体视觉技术的实现及其进展[J]. 电子技术应用, 2004, (10): 4-6.

[3] 王国美，陈孝威. SIFT特征匹配算法研究[J]. 盐城工学院学报(自然科学版), 2007, (02): 1-5.

[4] 张锐娟，张建奇，杨翠. 基于SURF的图像配准方法研究[J]. 红外与激光工程, 2009, 38(01): 160-165.

[5] 冯嘉. SIFT算法的研究和改进[D]. 吉林大学, 2010.

[6] 肖飞. 基于图像特征提取和特征点描述的匹配算法研究及其应用[D]. 电子科技大学, 2013.

[7] 索春宝，杨东清，刘云鹏. 多种角度比较SIFT、SURF、BRISK、ORB、FREAK算法[J]. 北京测绘, 2014, (04): 23-26.

[8] Hans-P Moravec. Towards Automatic Visual Obstacle Avoidance. 1977.

[9] M-J Stephens, Harris Christopher-G. 3D wire-frame integration from image sequences. 1989.

[10] G LoweDavid. Distinctive Image Features from Scale-Invariant Keypoints[J]. International Journal of Computer Vision, 2004.

[11] Herbert Bay, Tuytelaars Tinne, Van Gool Luc. SURF: Speeded Up Robust Features. 2006.

[12] Michael Calonder, Lepetit Vincent, Strecha Christoph, et al. BRIEF: Binary Robust Independent Elementary Features. 2010.

[13] Ethan Rublee, Rabaud Vincent, Konolige Kurt, et al. ORB: An efficient alternative to SIFT or SURF[J]. 2011 International Conference on Computer Vision, 2011, 2564-2571.

[14] Jian Zhao, Liu Hengzhu, Feng Yiliu, et al. BE-SIFT: A More Brief and Efficient SIFT Image Matching Algorithm for Computer Vision. 2015: 568-574.

[15] Fengjing Zhang, Fan Xiuying, Wang Zhiqiang, et al. Medium and low altitude CCD image Stitching based on SUFT algorithm. 2022: 1-4.

[16] 李小红，谢成明，贾易臻，等. 基于ORB特征的快速目标检测算法[J]. 电子测量与仪器学报, 2013, 27(05): 455-460.

[17] 梁焕青，范永弘，万惠琼，等. 一种运用AKAZE特征的无人机遥感影像拼接方法[J]. 测绘科学技术学报, 2016, 33(01): 71-75.

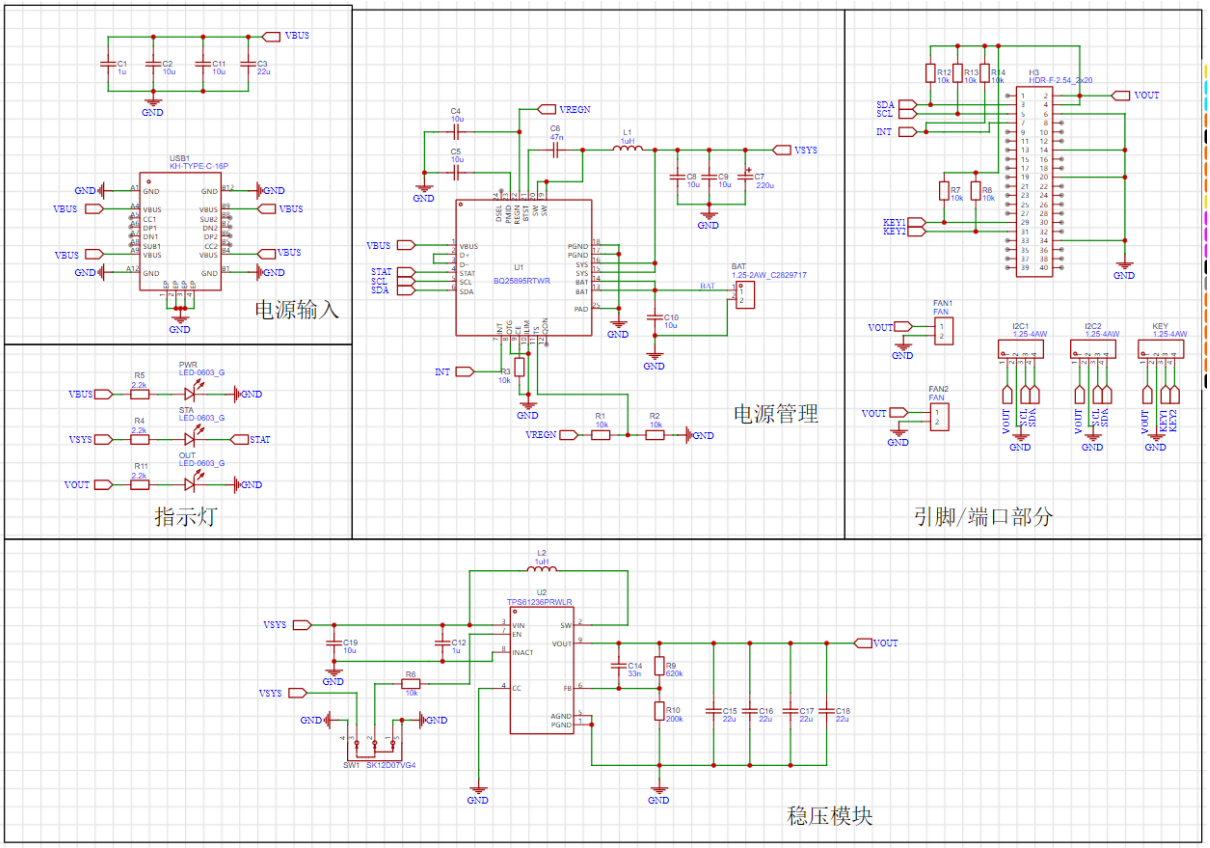
[18] Daniel DeTone, Malisiewicz Tomasz, Rabinovich Andrew. SuperPoint: Self-Supervised Interest Point Detection and Description[J]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, 337-33712.

[19] Eric Brachmann, Krull Alexander, Nowozin Sebastian, et al. DSAC — Differentiable RANSAC for Camera Localization[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 2492-2500.

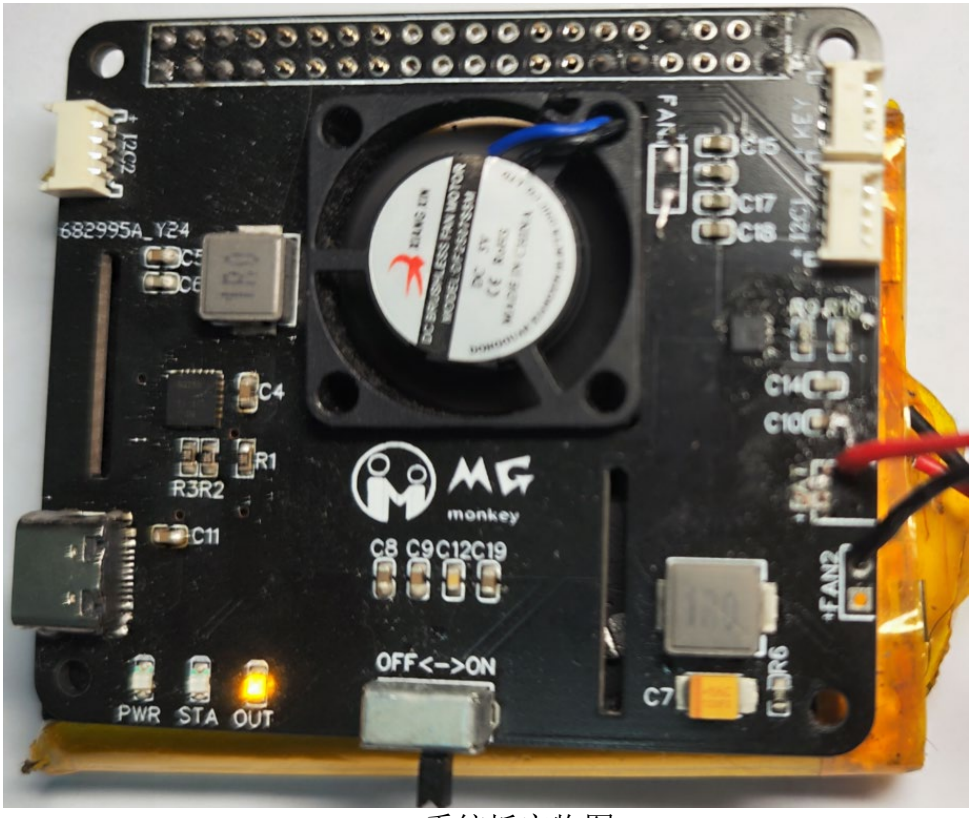
[20] Paul-Edouard Sarlin, DeTone Daniel, Malisiewicz Tomasz, et al. SuperGlue: Learning Feature Matching With Graph Neural Networks[J]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, 4937-4946.

[21] 支丹阳，张明友，张长均，等. 基于麒麟操作系统的碳硫分析软件开发设计[J]. 冶金分析, 2022, 42(11): 89-93.

附 录1



UPS 系统板原理图



UPS 系统板实物图

附 录 2

特征提取算法流程图如下：

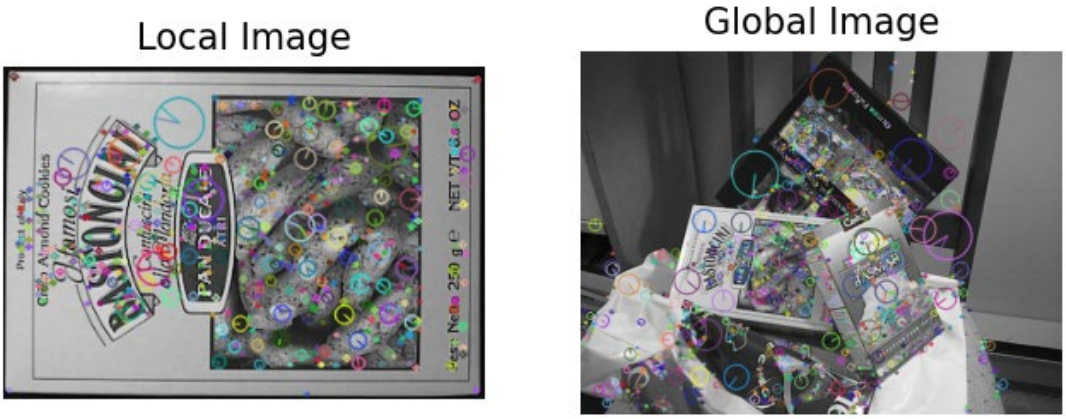


图1 特征提取

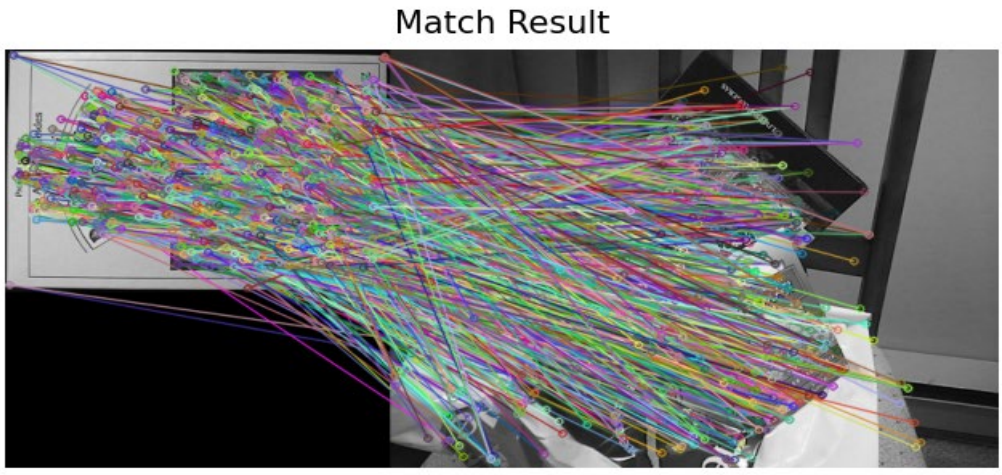


图2 基于FLANN算法匹配

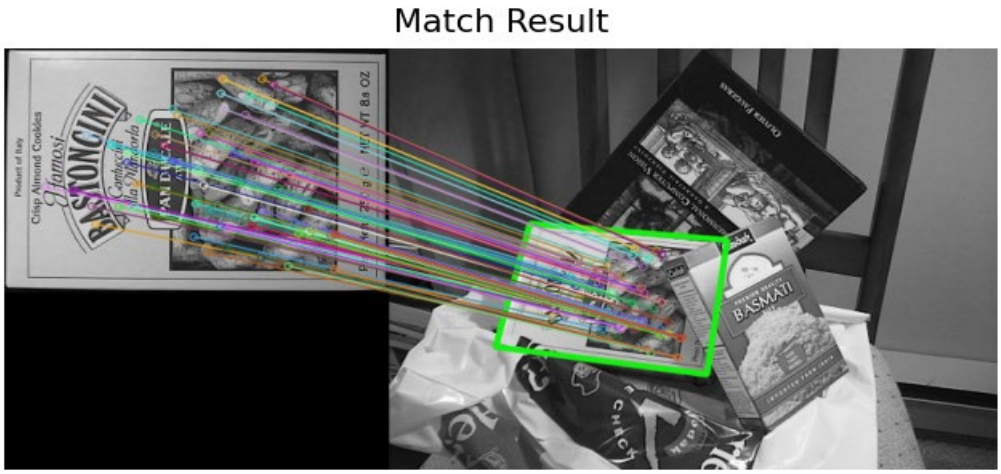


图3 基于RACSAC算法过滤匹配点

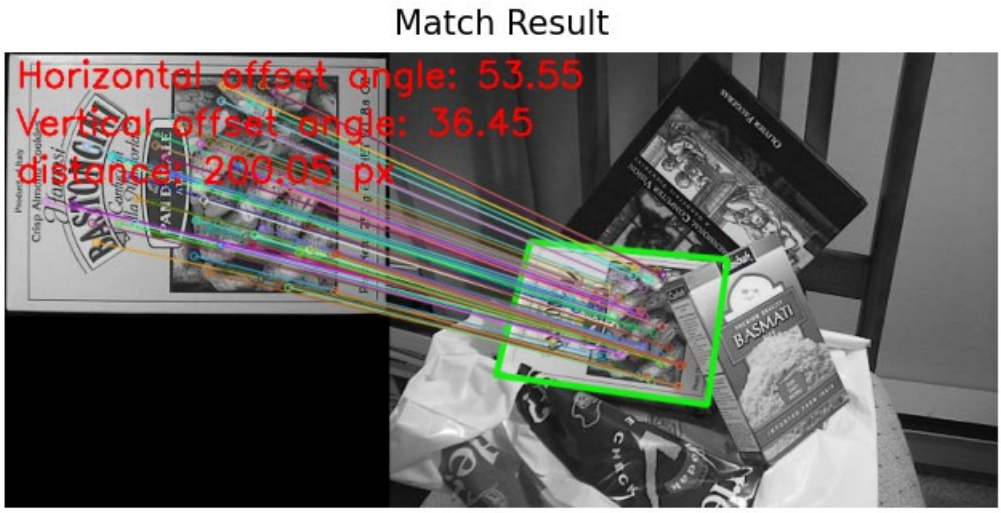
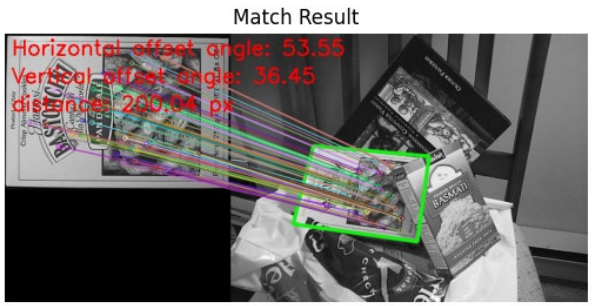


图4 根据匹配点计算偏移角和直线距离

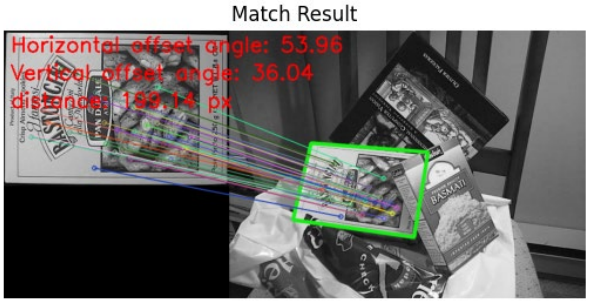
图5 软件自动汇总结果

附 录 3

基于各种特征匹配算法结果图：

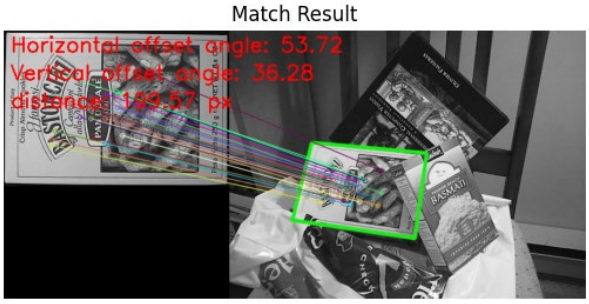


基于 SIFT 算法

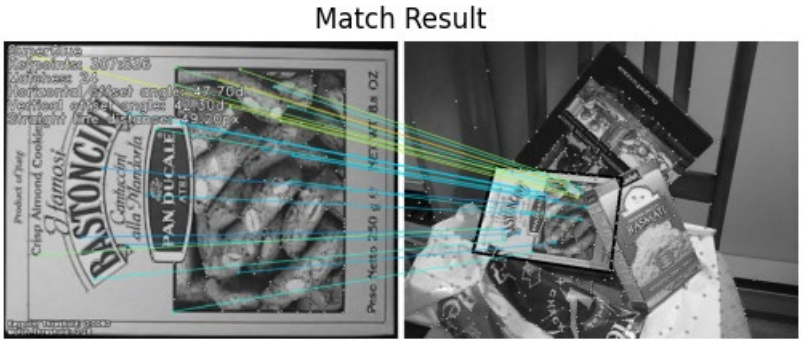


基于 SURF 算法

基于 ORB 算法



基于 AKAZE 算法



基于 SuperGlue 和 SuperPoint 算法

附 录 4

各算法部分核心代码如下：

```
global img_result, matchPointRate, mathTimeCost, HError, VError, DError, horizontal_angle,
vertical_angle, distance
img_result = None
# 读取原图像和局部图像
if global_img_path is not None and local_img_path is not None:
    img_global = cv2.imread(self.Settings.ImageChooseGloadPath)
    img_local = cv2.imread(self.Settings.ImageChooseLocalPath)
elif self.Settings.ImageChooseGloadPath != '' and self.Settings.ImageChooseLocalPath != '':
    img_global = cv2.imread(self.Settings.ImageChooseGloadPath)
    img_local = cv2.imread(self.Settings.ImageChooseLocalPath)
else:
    UIFunctions.showMessageBox(self, '请先选择图片！', 1)
    timer.stop()
    return False
##################################
#              1. 特征提取              #
##################################
img_global_gray= cv2.cvtColor(img_global, cv2.COLOR_BGR2GRAY)
img_local_gray= cv2.cvtColor(img_local, cv2.COLOR_BGR2GRAY)
if index == None:
    featureAlgorithm = int(self.Settings.FeatureAlgorithm)
else:
    featureAlgorithm = index
# SIFT 算法
if featureAlgorithm == 0:
    # 适配 OpenCV4.0
    if cv2.__version__.startswith('4.'):
        sift = cv2.SIFT_create()
    else:
        sift = cv2.xfeatures2d.SIFT_create()
    # 提取特征点和特征描述符
    kp_global, des_global = sift.detectAndCompute(img_global_gray, None)
    kp_local, des_local = sift.detectAndCompute(img_local_gray, None)
    FLANN_INDEX = 1
# SURF 算法
elif featureAlgorithm == 1:
    # 适配 OpenCV4.x
    if cv2.__version__.startswith('4.'):
        UIFunctions.showMessageBox(self, "安装的 Opencv 版本不包含 SURF 算法，建
议安装 3.4.1.15 版本！", 2)
        return False
```

```
            # surf = cv2.xfeatures2d.SURF_create(400, nOctaves=4, extended=True)
        else:
            surf = cv2.xfeatures2d.SURF_create(400, nOctaves=4, extended=True,
upright=True)

        # 在全景图和局部图中提取特征点和描述符
        kp_global, des_global = surf.detectAndCompute(img_global_gray, None)
        kp_local, des_local = surf.detectAndCompute(img_local_gray, None)
        # 使用 FLANN 匹配器进行特征点匹配
        FLANN_INDEX = 1
# ORB 算法
elif featureAlgorithm == 2:
        orb = cv2.ORB_create()
        kp_global, des_global = orb.detectAndCompute(img_global_gray, None)
        kp_local, des_local = orb.detectAndCompute(img_local_gray, None)
        FLANN_INDEX = 6
# AKAZE 算法
elif featureAlgorithm == 3:
        akaze = cv2.AKAZE_create()
        kp_global, des_global = akaze.detectAndCompute(img_global_gray, None)
        kp_local, des_local = akaze.detectAndCompute(img_local_gray, None)
        FLANN_INDEX = 6
# 基于 SuperPoint 和 SuperGlue 算法
elif featureAlgorithm == 4:
        imgMatch(self.Settings.ImageChooseLocalPath, self.Settings.ImageChooseGloadPath,
                 keypoint_threshold=self.Settings.KeypointThreshold,
match_threshold=self.Settings.MatchThreshold)
        return
# 描述提取的特征点
self.ui.btn_Setp1.setStyleSheet("border-color: rgb(239, 87, 119)")
self.ui.line1.setStyleSheet("color: rgb(255, 241, 0)")
img_global_keypoints = cv2.drawKeypoints(img_global, kp_global, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
img_local_keypoints = cv2.drawKeypoints(img_local, kp_local, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
if showLiucheng:
        timer.pause()
        UIFunctions.showImage(self, [img_local_keypoints, img_global_keypoints])
        timer.resume()
# 索引算法：0/1 :基于 KD 树       6：基于 LSH（局部敏感哈希）
if FLANN_INDEX == 0 or FLANN_INDEX == 1:
        index_params = dict(algorithm=FLANN_INDEX, trees=5)
        search_params = dict(checks=100)
        flann = cv2.FlannBasedMatcher(index_params, search_params)
```

```
        matches = flann.knnMatch(des_local, des_global, k=2)
elif FLANN_INDEX == 6:
        index_params = dict(algorithm=FLANN_INDEX, table_number=6, key_size=12,
multi_probe_level=1)
        search_params = dict(checks=100)
        flann = cv2.FlannBasedMatcher(index_params, search_params)
        matches = flann.knnMatch(des_local, des_global, k=2)

self.ui.btn_Setp2.setStyleSheet("border-color: #575fcf")
self.ui.line1.setStyleSheet("color: rgb(108, 252, 20)")
self.ui.line2.setStyleSheet("color: rgb(255, 241, 0)")
# 将匹配结果绘制到图像上
img_result = cv2.drawMatchesKnn(img_local, kp_local, img_global, kp_global, matches,
None, flags=2)
self.ui.label_matchPoints.setText(f"匹配点数：{len(matches)}/{len(kp_global)}")
if showLiucheng:
        timer.pause()
        UIFunctions.showImage(self, img_result)
        timer.resume()
# 应用尺度空间扩展和 RANSAC 算法来提高匹配准确性
good_matches = []
MIN_MATCH_COUNT = self.Settings.MatchDeadline
for m, n in matches:
        if m.distance < self.Settings.MatchRatio * n.distance:
                good_matches.append(m)
if len(good_matches) > MIN_MATCH_COUNT:
        # 获取关键点的坐标
        src_pts = np.float32([kp_local[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
        dst_pts = np.float32([kp_global[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)
        # 返回值中 M 为变换矩阵，也叫单应性矩阵
        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
        matchesMask = mask.ravel().tolist()
        h, w = img_local_gray.shape[:2]
        # 使用得到的变换矩 对原图像的四个角变换 获得在目标图像上对应的坐标
        pts = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1, 0]]).reshape(-1, 1, 2)
        dst = cv2.perspectiveTransform(pts, M)
        img_global = cv2.polylines(img_global, [np.int32(dst)], True, (17, 241, 21), 3,
cv2.LINE_AA)
else:
        matchesMask = None
self.ui.btn_Setp3.setStyleSheet("border-color: #4bcffa")
self.ui.line2.setStyleSheet("color: rgb(108, 252, 20)")
self.ui.line3.setStyleSheet("color: rgb(255, 241, 0)")
draw_params = dict(singlePointColor=None,
```

```
                    matchesMask=matchesMask,    # draw only inliers
                    flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
img_result = cv2.drawMatches(img_local, kp_local, img_global, kp_global, good_matches,
None, **draw_params)
label = [self.ui.label1_1_1, self.ui.label2_1_1, self.ui.label3_1_1, self.ui.label4_1_1,
self.ui.label5_1_1]
if index != None:
    label[index].setText(f"特征匹配点：{len(good_matches)}/{len(kp_global)}")
    matchPointRate[index] = len(good_matches) / len(kp_global)
else:
    label[featureAlgorithm].setText(f"特征匹配点：{len(good_matches)}/{len(kp_global)}")
    matchPointRate[featureAlgorithm] = len(good_matches) / len(kp_global)
    self.ui.label_matchPoints.setText(f"匹配点数：{len(good_matches)}/{len(kp_global)}")
if showLiucheng:
        UIFunctions.showImage(self, img_result)
# 创建一个浮点数数列
arr = np.array(matchPointRate)
# 找到最值的索引
print(matchPointRate)
label = [self.ui.label1_1_2, self.ui.label2_1_2, self.ui.label3_1_2, self.ui.label4_1_2,
self.ui.label5_1_2]
sort_list = np.argsort(arr)
label[sort_list[0]].setText("1")
label[sort_list[1]].setText("2")
label[sort_list[2]].setText("3")
label[sort_list[3]].setText("4")
label[sort_list[4]].setText("5")
# 计算局部图像和全局图像之间的变换矩阵
h, w = img_local.shape[:2]
corners_global = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1, 0]]).reshape(-1, 1, 2)
corners_local = cv2.perspectiveTransform(corners_global, M)
# 根据变换矩阵计算水平角、垂直角和直线距离
dx = corners_local[0][0][0] - corners_global[0][0][0]
dy = corners_local[0][0][1] - corners_global[0][0][1]
horizontal_angle = np.rad2deg(np.arctan2(dy, dx))
vertical_angle = np.rad2deg(np.arctan2(dx, dy))
distance = np.sqrt(dx**2 + dy**2)

self.ui.btn_Setp4.setStyleSheet("border-color: #0be881")
self.ui.line3.setStyleSheet("color: rgb(108, 252, 20)")
self.ui.line4.setStyleSheet("color: rgb(255, 241, 0)")
cv2.putText(img_result, u"Horizontal offset angle: {:.2f}".format(horizontal_angle), (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
```

```
cv2.putText(img_result, u"Vertical offset angle: {:.2f}".format(vertical_angle), (10, 70),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
self.ui.label_HOffset.setText("水平偏移角：{:.2f}° ".format(horizontal_angle))
self.ui.label_VOffset.setText("垂直偏移角：{:.2f}° ".format(vertical_angle))
if index != None:
    label = [self.ui.label1_H, self.ui.label2_H, self.ui.label3_H, self.ui.label4_H,
self.ui.label5_H]
    label[index].setText("水平偏移角：{:.2f}° ".format(horizontal_angle))
    label = [self.ui.label1_V, self.ui.label2_V, self.ui.label3_V, self.ui.label4_V,
self.ui.label5_V]
    label[index].setText("垂直偏移角：{:.2f}° ".format(vertical_angle))
else:
    label = [self.ui.label1_H, self.ui.label2_H, self.ui.label3_H, self.ui.label4_H,
self.ui.label5_H]
    label[featureAlgorithm].setText("水平偏移角：{:.2f}° ".format(horizontal_angle))
    label = [self.ui.label1_V, self.ui.label2_V, self.ui.label3_V, self.ui.label4_V,
self.ui.label5_V]
    label[featureAlgorithm].setText("垂直偏移角：{:.2f}° ".format(vertical_angle))
if showLiucheng:
    timer.pause()
    UIFunctions.showImage(self, img_result)
    timer.resume()
# 计算误差
if float(self.Settings.ResultHAngle) != 0:
    label = [self.ui.label1_HError, self.ui.label2_HError, self.ui.label3_HError,
self.ui.label4_HError, self.ui.label5_HError]
    if index != None:
        Camera.calcError(self, horizontal_angle, self.Settings.ResultHAngle,
self.ui.label_HError, label[index])
    else:
        Camera.calcError(self, horizontal_angle, self.Settings.ResultHAngle,
self.ui.label_HError, label[featureAlgorithm])
if float(self.Settings.ResultVAngle) != 0:
    label = [self.ui.label1_VError, self.ui.label2_VError, self.ui.label3_VError,
self.ui.label4_VError, self.ui.label5_VError]
    if index != None:
        Camera.calcError(self, vertical_angle, self.Settings.ResultVAngle,
self.ui.label_VError, label[index])
    else:
        Camera.calcError(self, vertical_angle, self.Settings.ResultVAngle,
self.ui.label_VError, label[featureAlgorithm])
Camera.calcHorizontalDistance(self)
self.ui.btn_Setp5.setStyleSheet("border-color: #f8a5c2")
self.ui.line4.setStyleSheet("color: rgb(108, 252, 20)")
```

```
cv2.putText(img_result, "Straight line distance: {:.2f} px".format(distance), (10, 110),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
self.ui.label_HDistance.setText("直线距离：{:.2f} px".format(distance))
if index != None:
    label = [self.ui.label1_D, self.ui.label2_D, self.ui.label3_D, self.ui.label4_D,
self.ui.label5_D]
    label[index].setText("直线距离：{:.2f}px".format(distance))
else:
    label = [self.ui.label1_D, self.ui.label2_D, self.ui.label3_D, self.ui.label4_D,
self.ui.label5_D]
    label[featureAlgorithm].setText("直线距离：{:.2f}px".format(distance))
timer.pause()
UIFunctions.showImage(self, img_result)
timer.stop()
# 显示耗时
label = [self.ui.label1_2_1, self.ui.label2_2_1, self.ui.label3_2_1, self.ui.label4_2_1,
self.ui.label5_2_1]
print("{}耗时：{:.2f} s".format(self.Settings.getFeatureAlgorithm(),
timer.get_elapsed_time()))
self.ui.label_timeCount.setText("耗时：{:.2f} s".format(timer.get_elapsed_time()))
if index != None:
    label[index].setText("耗时：{:.2f} s".format(timer.get_elapsed_time()))
    label[index].setText("耗时：{:.2f} s".format(timer.get_elapsed_time()))
    mathTimeCost[index] = timer.get_elapsed_time()
else:
    label[featureAlgorithm].setText("耗时：{:.2f} s".format(timer.get_elapsed_time()))
    label[featureAlgorithm].setText("耗时：{:.2f} s".format(timer.get_elapsed_time()))
    mathTimeCost[featureAlgorithm] = timer.get_elapsed_time()
# 创建一个浮点数数列
arr = np.array(mathTimeCost)
# 找到最值的索引
print(mathTimeCost)
sort_list = np.argsort(arr)
label = [self.ui.label1_2_2, self.ui.label2_2_2, self.ui.label3_2_2, self.ui.label4_2_2,
self.ui.label5_2_2]
label[sort_list[0]].setText("1")
label[sort_list[1]].setText("2")
label[sort_list[2]].setText("3")
label[sort_list[3]].setText("4")
label[sort_list[4]].setText("5")
# 总体评分
all_data = [x + y for x, y in zip(matchPointRate, mathTimeCost)]
arr = np.array(all_data)
sort_list = np.argsort(arr)
```

```
label = [self.ui.label1_3, self.ui.label2_3, self.ui.label3_3, self.ui.label4_3, self.ui.label5_3]
label[sort_list[0]].setText("最好")
label[sort_list[1]].setText("较好")
label[sort_list[2]].setText("一般")
label[sort_list[3]].setText("较差")
label[sort_list[4]].setText("最差")
```