基于AutoStitch的无人机航拍图像拼接技术研究

黄锦波

目录

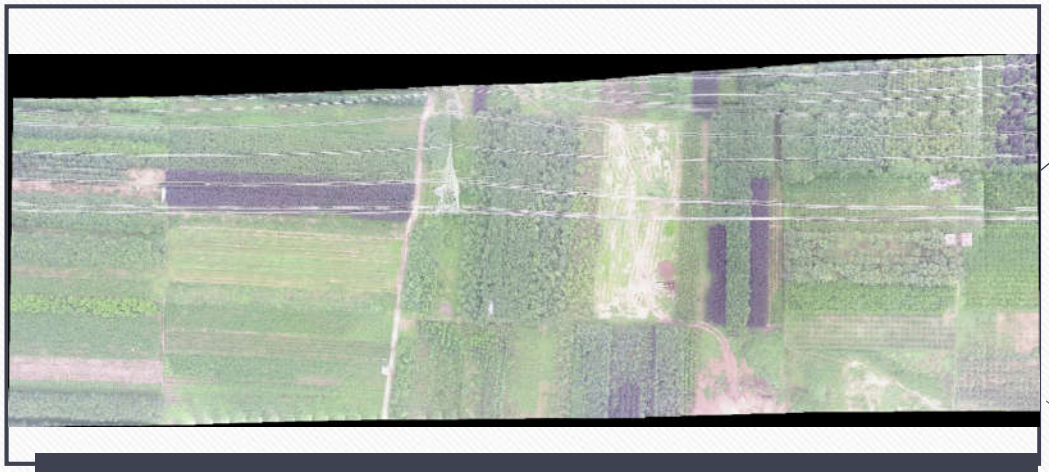
研究背景及意义

1 选题背景

2 国内外研究现状

3 算法简析与选型

01 研究背景及意义—选题背景



宽视角地理信息拼接结果图片

拍摄地点：西安郊区

原始图像分辨率：6002*2163

无人机航拍

(数目50+甚至100+)

航拍图像拼接

A5

01 研究背景及意义—国内外研究现状

基于已知相机位置的图像拼接

多个摄像头

预计算变换矩阵

视频拼接

实时拼接

AutoStitch算法

基于特征点检测

建立相机模型

计算单应性矩阵

光束法平差

APAP拼接算法

基于特征点检测

密集网络划分

多个单应性矩阵

基于无人机POS信息的拼接算法

基于无人机空间位置

POS信息获取传输

坐标转换

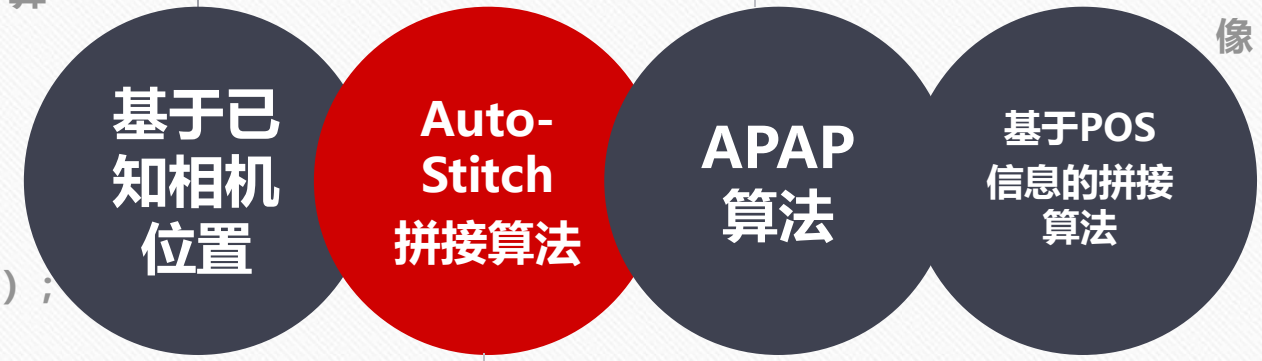
幻灯片 5

A5 红色表示已知相机位置，蓝色是未知相机位置

Administrator, 2020/11/9 星期一

A4

01 研究背景及意义—算法简析与选型



优点：

实时拼接，适合视频拼接。

缺点：

需要多个相机，并事先计算单应性矩阵。

优点：

拼接图像多（论文中20+）；

每个步骤技术选型多；

现有研究和开源资料多。

缺点：

拼接结果容易产生错位和鬼影。

选用AutoStitch算法作为基础算法进行改进。

优点：

拼接效果好，鬼影和错位少。

缺点：

开源资源少，缺乏C++实现；

拼接数目少（论文中2张图像）。

优点：

边拍边拼；

不需要建模计算变换矩阵；

拼接数目多；

缺点：

缺少论文资料和开源资源；

需要坐标转换；

无人机POS信息不易获取。

幻灯片 6

A4 红色标识AutoStitch被选中

Administrator, 2020/11/9 星期一

1 图像拍摄和输入

2 图像配准

3 图像投影

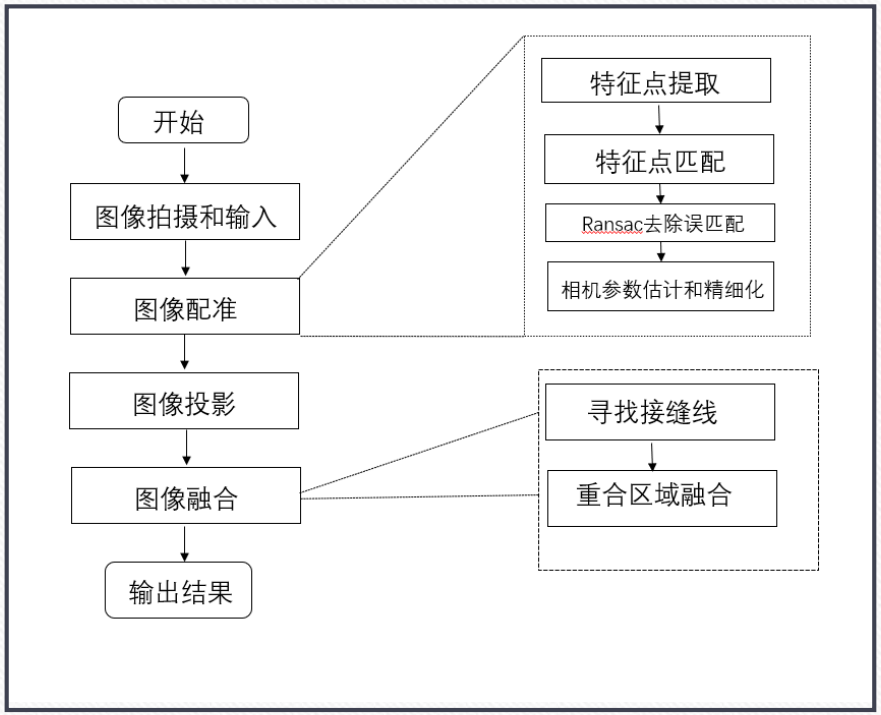
4 图像融合

西安电子科技大学
XIDIAN UNIVERSITY

总体算法流程

02 总体算法流程与技术选型

AutoStitch算法流程



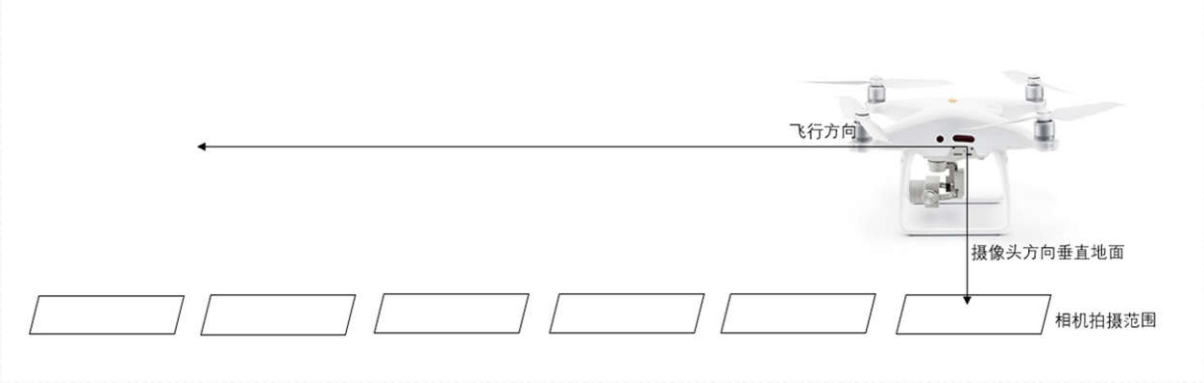
测试条件

软件库：OpenCV 3.4.3+OpenCV Contrib 3.4.3
硬件环境：Windows10，16G内存，Inteli59300h CPU，NVIDIA GeForce GTX1660Ti 显卡，6G 显存
航拍无人机：大疆精灵4

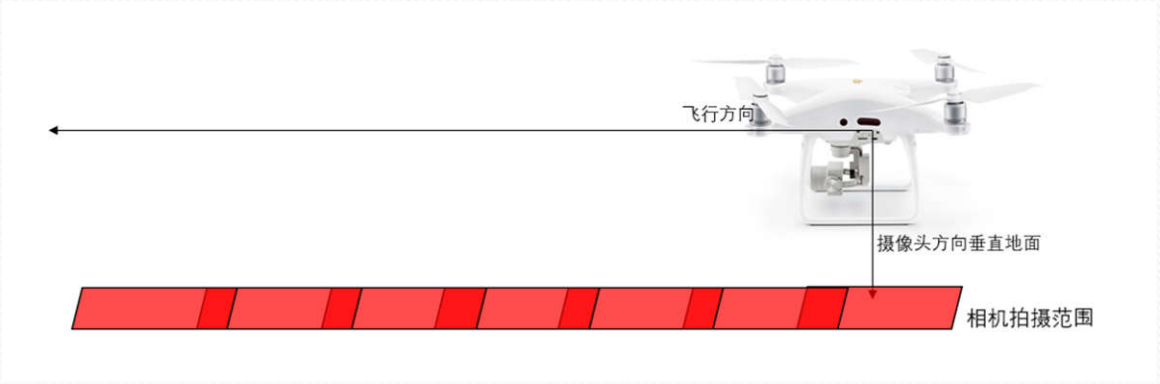
A3

02 总体算法流程与技术选型—图像拍摄与输入

无人机飞行方向与摄像头方向



图像重叠情况



图像序列获取流程

单向飞行，获取视频

↓

设置采样率

↓

设置命名格式

↓

视频采样，命名图像

幻灯片 9

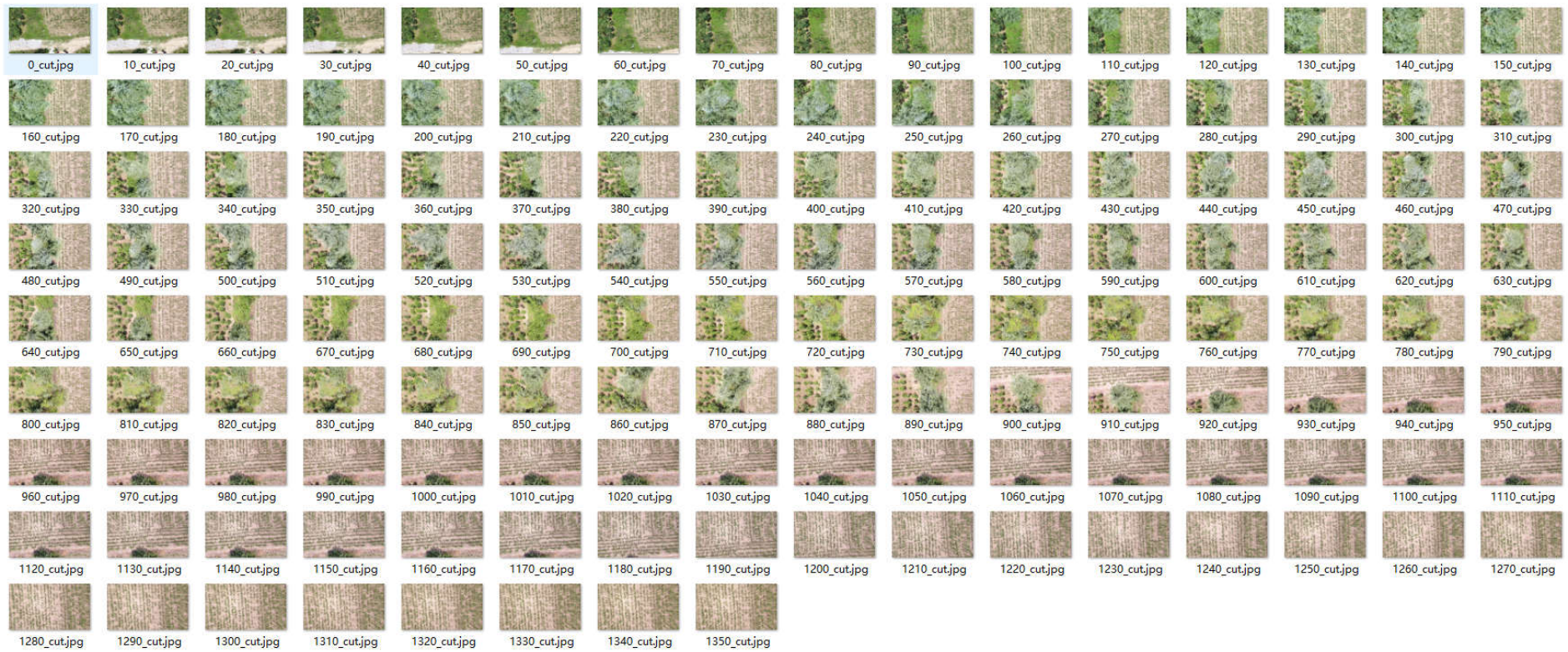
A3 设置命名格式是为了后续的图像拼接软件能够自动排序

Administrator, 2020/11/9 星期一

02 总体算法流程与技术选型—图像拍摄与输入

图像序列示意

拍摄于西安郊区，采样率为10（隔10帧采样）



02 总体算法流程与技术选型—图像配准

特征点检测

Sift特征点检测

特征点耗时

图像数目	10	20	30	40	50
特征点检测时间(sec)	5.74841 s	12.1992	18.7121	25.2284	32.0666
图像数目	60	70	80	90	100
特征点检测时间(sec)	38.5154	44.6146	54.0255	61.1873	68.486

A6

02 总体算法流程与技术选型—图像配准

Ransac去除误匹配

Ransac去除误匹配

图像投影

选用平面投影。投影公式为：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R^{-1}K^{-1}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

投影逆向公式为：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = KR\begin{bmatrix} \frac{u}{s} - t_1 \\ \frac{v}{s} - t_2 \\ 1 - t_3 \end{bmatrix}$$

幻灯片 12

A6 去除误匹配之前需要特征点匹配，使用BBF匹配

Administrator, 2020/11/9 星期一

02 总体算法流程与技术选型—图像融合

寻找图像接缝线

接缝线掩膜

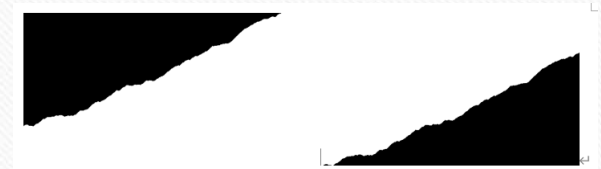
拼接结果图

使用三组图像测试接缝线算法,其中一组测试图片（两张）：

Voronoi

动态规划

图割法



三种接缝线算法耗时：

算法	Voronoi	动态规划法	图割法
第一组耗时（sec）	0.0380089	0.212062	15.7863
第二组耗时（sec）	0.0178526	0.169147	8.21235
第三组耗时（sec）	0.029053	0.196751	29.7938

关键技术研究与验证

1 拼接耗时统计

2 光束法平差改进

3 内存分析与改进

4 多线程加速

03 关键技术研究与验证—拼接耗时统计

待拼接图像序列

各组拼接大小与耗时

图像拼接数目	10	20	30	40	50
拼接耗时（sec）	36.5828	73.787	173.736	318.479	497.48
拼接结果大小（像素）	1926*1281	1951*2077	2009*2912	2029*3664	2053*4533
图像拼接数目	60	70	80	90	100
拼接耗时（sec）	761.566	1122.73	1648.98	2277.08	3128.1
拼接结果大小（像素）	2074*5471	2223*6518	2421*7410	2408*7636	2453*8278

03 关键技术研究与验证—拼接耗时统计

拼接结果图（100张）

Sift特征点提取特征;

平面投影;

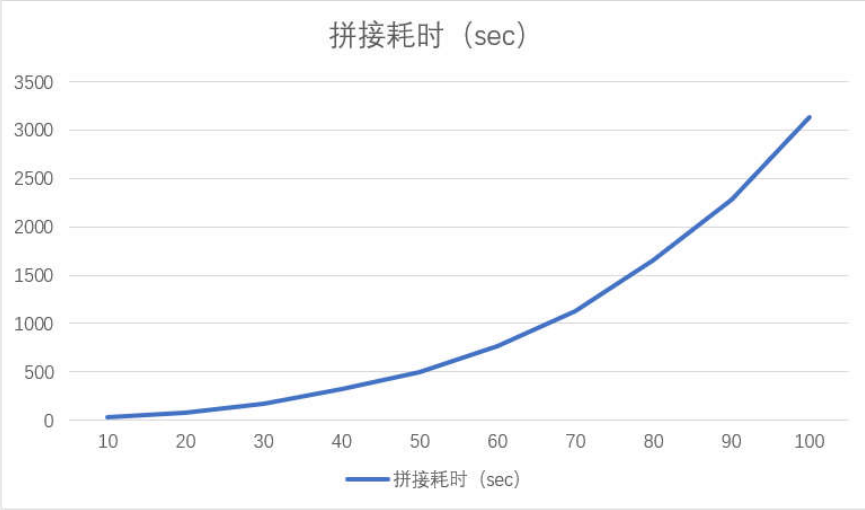
Voronoi接缝线算法;

融合方式采用多层融合

大小：2453*8278

耗时：3128.1sec

拼接耗时折线图



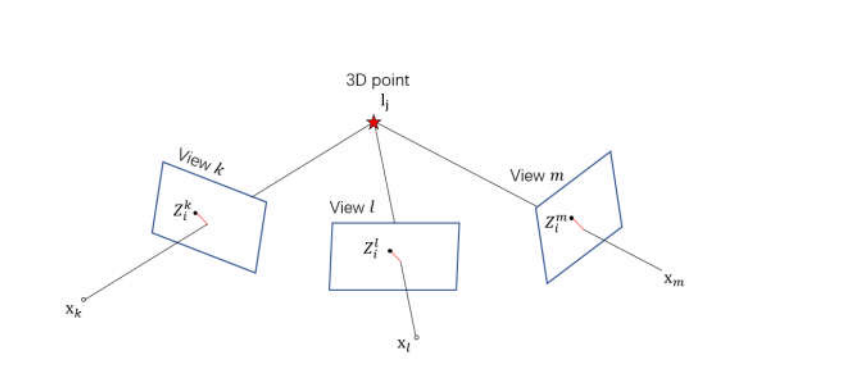
03 关键技术研究与验证—光束法平差改进

光束法平差耗时占比(10-100张)

图像拼接数目	10	20	30	40	50
拼接耗时（sec）	36.5828	73.787	173.736	318.479	497.48
光束法平差耗时（sec）	16.1983	37.414	119.866	248.773	409.77
光束法平差占比	44.28%	50.71%	68.9%	78.1%	82.37%

图像拼接数目	60	70	80	90	100
拼接耗时（sec）	761.566	1122.73	1648.98	2277.08	3128.1
光束法平差耗时（sec）	666.957	1007.46	1512.54	2127	2955.12
光束法平差占比	87.57%	89.73%	91.73%	93.41%	94.47%

光束法平差示意图



03 关键技术研究与验证—光束法平差改进

光束法平差步骤

1.对n 张图像参与拼接，建立一个有 n*4 行的列向量，保存四个变量（焦距，对X，Y，Z轴的旋转量）；
2.遍历图像序列，将每个图像的内参矩阵和外参矩阵保存到步骤1的变量中；
3.统计配对图像的匹配的特征点数目 matches（用于生成步骤4的雅克比矩阵）；
4.生成雅可比矩阵，以最小射线误差为目标，使用 LM 算法优化步骤进行迭代。

使用光束法平差（BA）与否对比

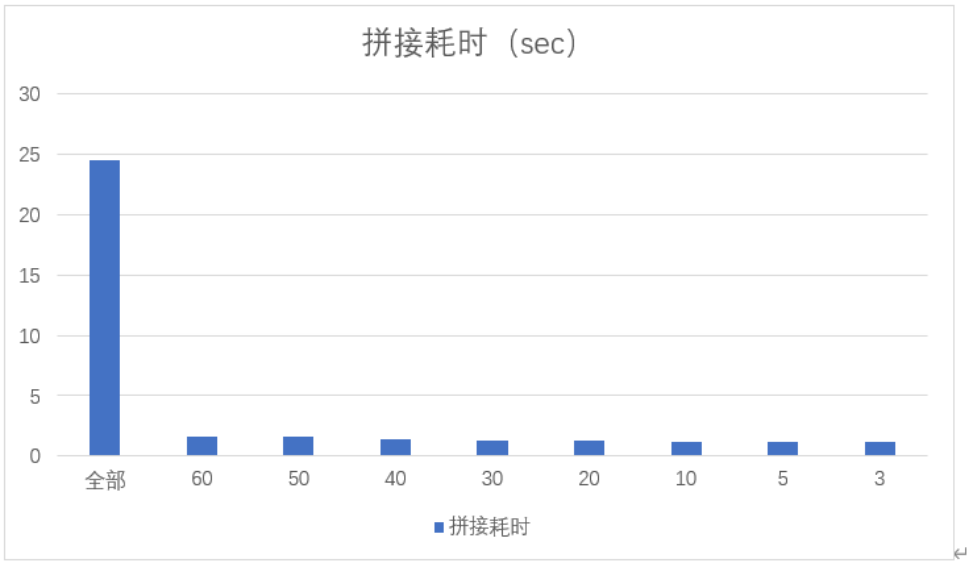
① 未使用BA（未还原真实场景）

✔使用了BA（还原了真实场景）

03 关键技术研究与验证—光束法平差改进

匹配点对优化

不同匹配点对耗时



不同匹配点对拼接结果对比

✔使用所有匹配点对拼接结果

✔使用60个匹配点对拼接结果

03 关键技术研究与验证—光束法平差改进

LM算法迭代次数

拼接图像数目	30	40	50	60	70
LM迭代次数	2996	2998	3000	2993	2998

OpenCV LM算法跳出条件

1. 误差矩阵err为0矩阵；（很难达到）

2.当前雅克比矩阵和上一次迭代的雅克比矩阵绝对范数或者相对范数小于2.2204460492503131e-016(这个值在OpenCV中由宏定义DBL_EPSILON定义)；（很难达到）

3.光束法平差过程中的雅可比矩阵有效迭代次数（非总迭代次数）大于1000。（大部分在此跳出）

03 关键技术研究与验证—光束法平差改进

射线误差下降情况

100张图

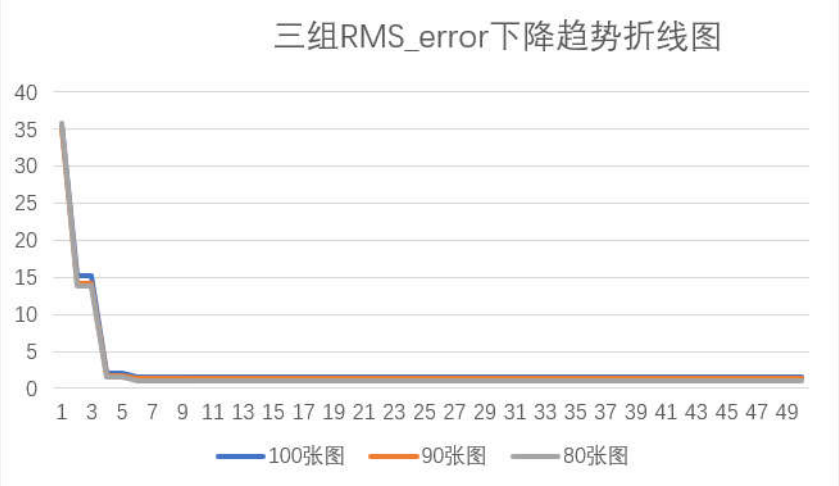
迭代次数	1	2	3	4	5	6
RMS_error	35.4904	15.2653	15.2653	2.04033	2.04033	1.49314
迭代次数	7	8	9	10	11-20	100+
RMS_error	1.49314	1.48727	1.48727	1.4871	1.4871	1.4871

90张图

迭代次数	1	2	3	4	5	6
RMS_error	35.0285	14.2447	14.2447	1.76934	1.76934	1.33733
迭代次数	7	8	9	10	11-20	100+
RMS_error	1.33733	1.33241	1.33241	1.33233	1.33233	1.33233

80张图

迭代次数	1	2	3	4	5	6
RMS_error	35.744	13.7589	13.7589	1.50447	1.50447	1.07703
迭代次数	7	8	9	10	11-20	100+
RMS_error	1.07703	1.07196	1.07196	1.07195	1.07195	1.07195



03 关键技术研究与验证—光束法平差改进

改进思路

1.对n 张图像参与拼接，建立一个有 n*4 行的列向量，保存四个变量（焦距，对X，Y，Z轴的旋转量）；
2.遍历图像序列，将每个图像的内参矩阵和外参矩阵保存到步骤1的变量中；
3.统计配对图像的匹配的特征点数目 matches（用于生成步骤4的雅克比矩阵）；
4.以最小射线误差为目标，使用 LM 算法优化步骤进行迭代。

步骤四的LM迭代中，如果迭代次数超过5次，计算最近5次的射线误差的方差。如果方差低于1e-5，跳出循环；否则，重新进行迭代。

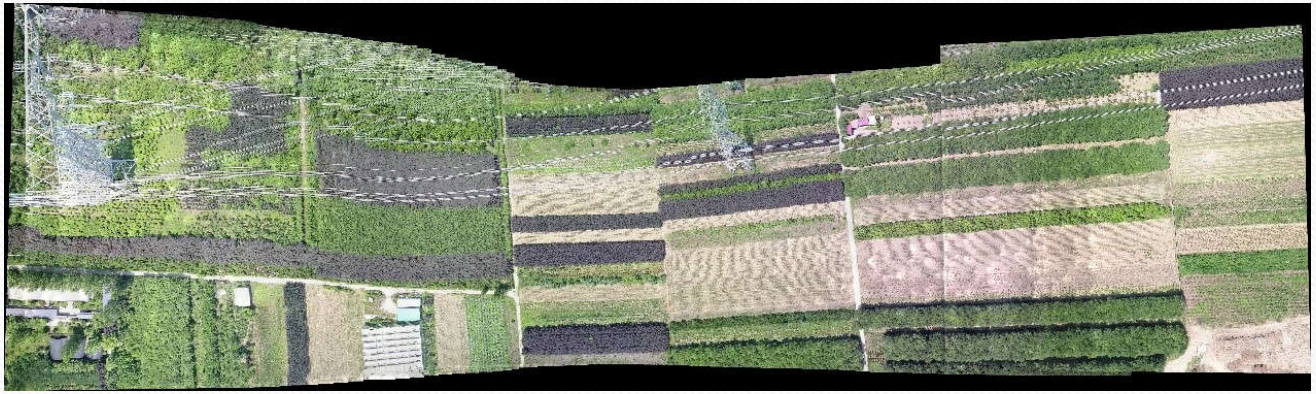
改进前后的结果图对比

✔改进前60张拼接结果图

✔改进后60张拼接结果图

03 关键技术研究与验证—光束法平差改进

改进前后的结果图对比



✔改进前100张拼接结果图

✔改进后100张拼接结果图

拼接结果相似度

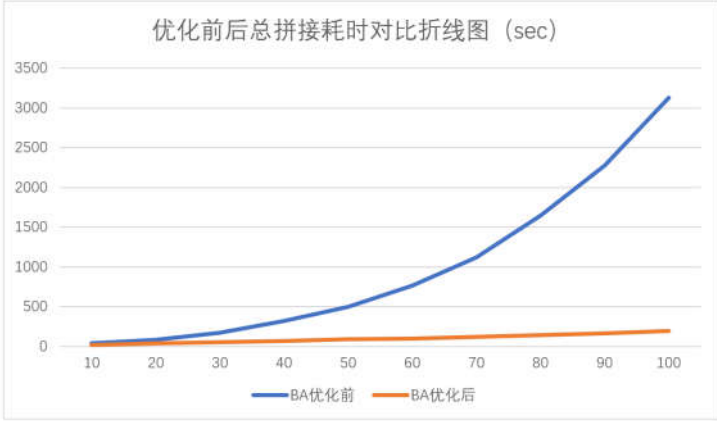
拼接数量	10	20	30	40	50
直方图相似度	0.999685	0.998899	0.999604	0.99998	0.99999
拼接数量	60	70	80	90	100
直方图相似度	0.99999	0.99974	0.99975	0.99987	0.999924

03 关键技术研究与验证—光束法平差改进

耗时对比

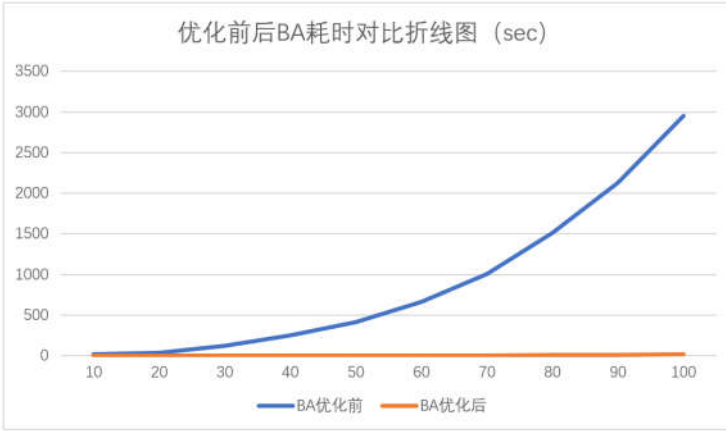
10-50张图

图像拼接数目	10	20	30	40	50
优化前拼接耗时（sec）	36.5828	73.787	173.736	318.479	497.48
优化前BA耗时（sec）	16.1983	37.414	119.866	248.773	409.77
优化后拼接耗时（sec）	20.1397	36.3004	54.3587	71.0453	89.7427
优化后BA耗时（sec）	0.0588675	0.192792	0.472842	1.18348	1.86798



60-100张图

图像拼接数目	60	70	80	90	100
优化前拼接耗时（sec）	761.566	1122.73	1648.98	2277.08	3128.1
优化前BA耗时（sec）	666.957	1007.46	1512.54	2127	2955.12
优化后拼接耗时（sec）	99.1455	119.768	145.818	163.839	193.39.1
优化后BA耗时（sec）	4.37548	4.9183	9.17972	13.2542	19.3183

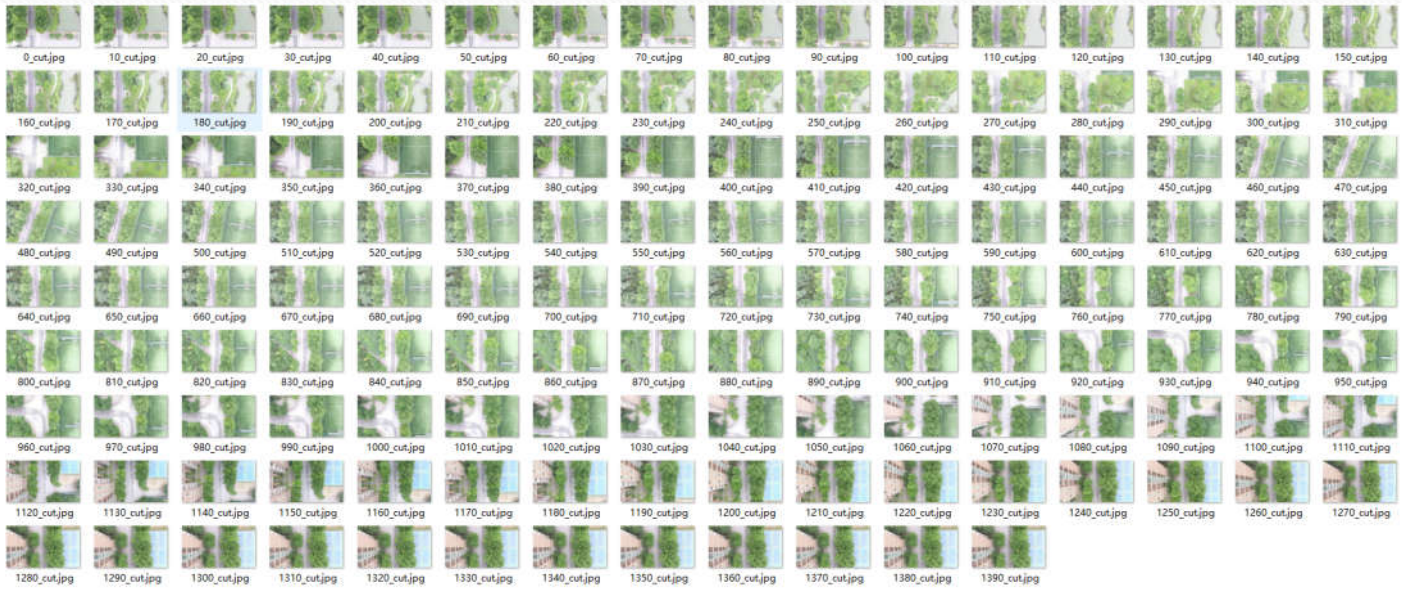


03 关键技术研究与验证—内存优化

实验中的内存报错信息

OpenCV Error: Assertion failed (clEnqueueReadBuffer_pfn(q, (cl_mem)u->handle, 1, 0, u->size, alignedPtr.getAlignedPtr(), 0, 0, 0) == 0) in cv::ocl::OpenCLAllocator::map, file D:\opencv-331\opencv\sources\modules\core\src\ocl.cpp, line 3796

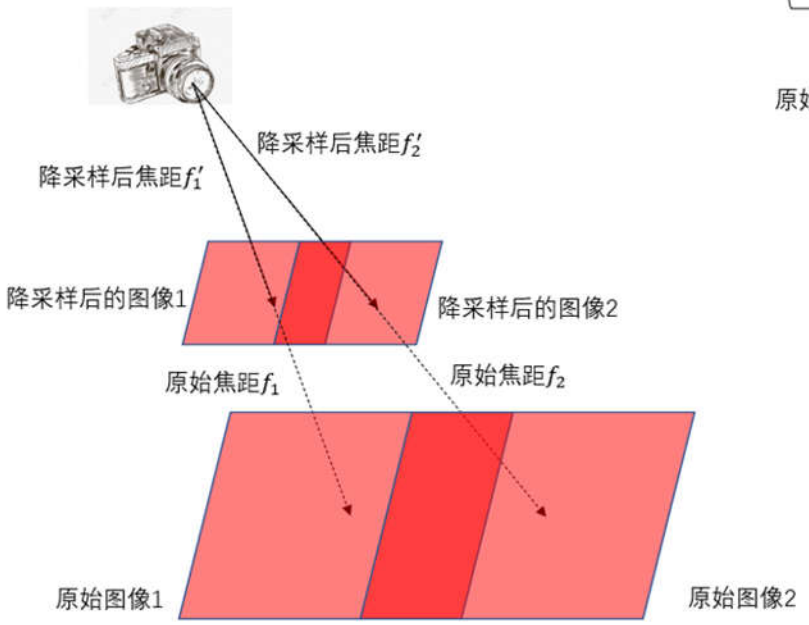
因内存报错不能拼接的140张图拼接实验



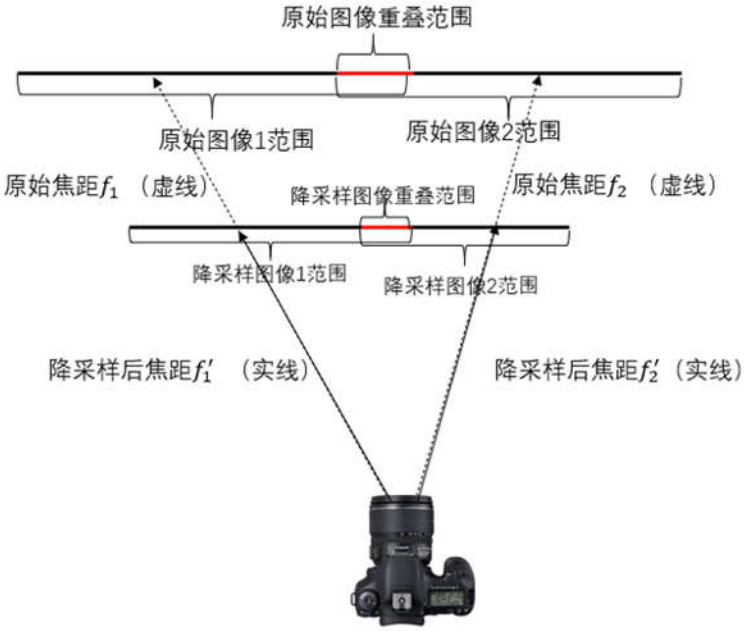
大多在特征点提取阶段报错。
主要原因：
特征点数目太多，导致：
1. 特征点描述子占用空间过大；
2. 建立高斯金字塔消耗内存过大

03 关键技术研究与验证—内存优化

降采样原理图



降采样和原始图变化关系示意图



降采样相似三角形示意图

03 关键技术研究与验证—内存优化

基于降采样的改进图像拼接流程

输入图像序列

↓

若像素大于10^6，则进行降采样,否则不做改变

↓

若降采样，释放旧图像，使用降采样后图像，否则不做改变

↓

特征点提取、匹配、光束法平差优化、图像投影等

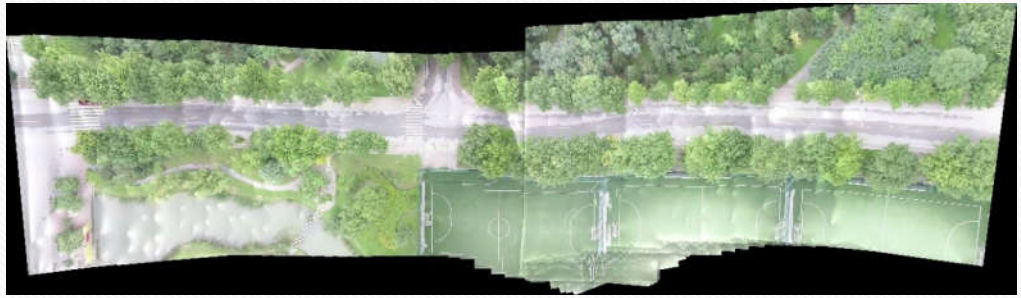
↓

重新输入图像，根据降采样比例恢复焦距、左上角点坐标等数据

↓

图像融合生成拼接结果图

改进前后拼接结果对比



✔改进前90张拼接结果图

✔改进后90张拼接结果图

✔改进后140张拼接结果图

03 关键技术研究与验证—内存优化

内存优化拼接信息

90张图拼接对比

算法	完全不降采样	先降采样后恢复
降采样前的图像尺寸	1920*1080	1920*1080
降采样后的图像尺寸	无	1568*882
平均图像特征点数	约21000个	约13000个
峰值内存使用	5.11Gb	2.21Gb
平均内存使用	3.8Gb	2.1G
拼接耗时（sec）	137.98	94.83
拼接结果大小（像素）	2564*8860	2971*10796

140张图拼接信息

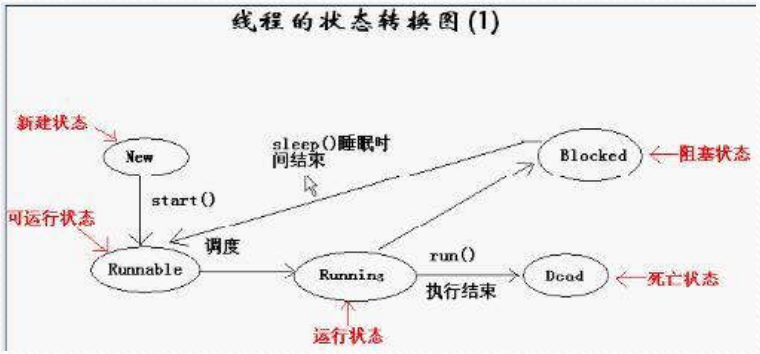
降采样前的图像尺寸	1920*1080
降采样后的图像尺寸	1568*882
平均图像特征点数	约13000个
降采样后峰值内存使用	3.55G
平均内存使用	3.18
拼接耗时	204.37sec
拼接结果大小	4825*19752

60-100张图拼接信息

图像拼接数目	60	70	80	90	100
第二章原始算法耗时(sec)	761.566	1122.73	1648.98	2277.08	3128.1
第三章算法拼接耗时（sec）	99.1455	119.768	145.818	163.839	193.39.1
降采样改进算法耗时（sec）	56.544	67.8762	80.2647	93.5783	118.147
第三章算法峰值内存（sec）	3.4223	3.8121	4.1325	4.3574	4.7425
降采样改进算法峰值内存（sec）	2.6628	2.8475	2.9693	3.1949	3.3687

03 关键技术研究与验证—多线程加速

线程状态示例图



OpenMP代码示例

```
#include <stdio.h>
#include <iostream>
#include <omp.h>
using namespace::std;
int main(int argc, char* argv[])
{
#pragma omp parallel for
    for (int i = 0; i < 10; i++)
    {

        printf("i = %d, I am  Thread :: %d\n",i,omp_get_thread_num());
    }
    return 0;
}
```

OpenMP输出结果

```
Microsoft Visual Studio 调试控制台
i = 0, I am  Thread :: 0
i = 1, I am  Thread :: 0
i = 7, I am  Thread :: 5
i = 8, I am  Thread :: 6
i = 9, I am  Thread :: 7
i = 4, I am  Thread :: 2
i = 2, I am  Thread :: 1
i = 3, I am  Thread :: 1
i = 6, I am  Thread :: 4
i = 5, I am  Thread :: 3
```

03 关键技术研究与验证—多线程加速

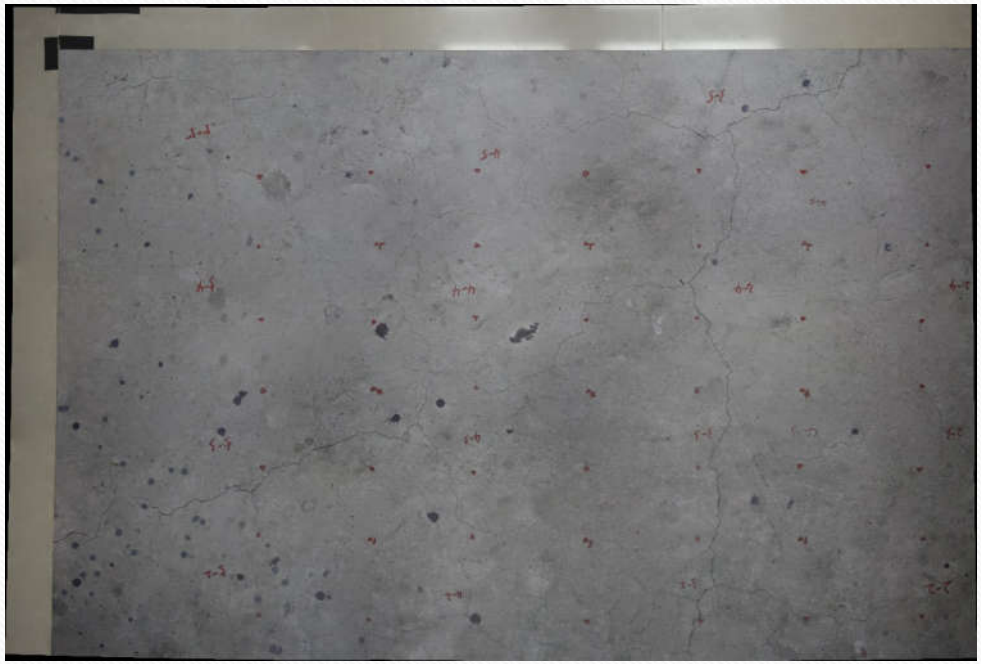
特征点检测多线程加速对比

140张图片加速前后特征点提取时间对比

组别	1	2	3	4	5
串行(sec)	44.1225	44.306	44.4613	44.4879	44.4601
8线程并行(sec)	25.6798	27.1618	27.0928	28.1246	27.5037

200张图片加速前后特征点提取时间对比

组别	1	2	3	4	5
串行(sec)	164.95	142.447	144.751	159.815	160.316
8线程并行(sec)	50.5295	50.4752	50.8589	51.5818	50.2379



原始大图（12526*8454）
滑动采样
获取200张图像测试
（5000*4000）
为了减少内存消耗，
只进行特征点检测，没有
特征点描述子保存；

03 关键技术研究与验证—多线程加速

多层融合步骤

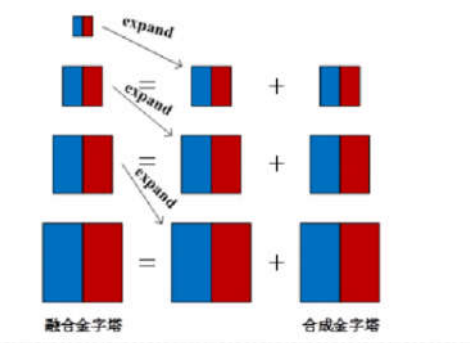
1.对于所有待拼接图像，建立拉普拉斯金字塔;

2.对拉普拉斯金字塔中的不同区域的不同层进行合并，得到合并金字塔;

3.对合并后的金字塔进行逆拉普拉斯变换，得到融合金字塔（底层即最终拼接结果）。

相互独立，可并行加速

多层融合示意图



融合阶段多线程加速对比

140张图片OpenMP加速多层融合对比

组别	1	2	3	4	5
串行（sec）	42.4986	44.889	45.4363	45.1988	45.5782
8线程并行（sec）	29.1572	29.8474	29.5758	29.4276	29.2312

200张图片OpenMP加速多层融合对比

组别	1	2	3	4	5
串行(sec)	399.685	412.635	396.635	411.697	410.289
8线程并行(sec)	254.658	251.689	254.29	253.269	252.14

03 关键技术研究与验证—多线程加速

各阶段拼接耗时对比

图像拼接数目	60	70	80	90	100
原始算法耗时(sec)	761.566	1122.73	1648.98	2277.08	3128.1
光束法平差优化算法拼接耗时（sec）	99.1455	119.768	145.818	163.839	193.39.1
降采样改进算法耗时（sec）	56.544	67.8762	80.2647	93.5783	118.147
多线程加速后耗时（sec）	41.9224	51.0393 sec	62.2513	72.8184	88.2915

动态规划(DP)找接缝线结果示例

DP与Voronoi接缝线耗时

图像拼接数目	60	70	80	90	100	140
Voronoi法耗时（sec）	41.9224	51.0393 sec	62.2513	72.8184	88.2915	120.284
DP法耗时(sec)	45.0051	54.8132	67.0965	78.5023	96.1874	142.653

1 拼接软件设计实现

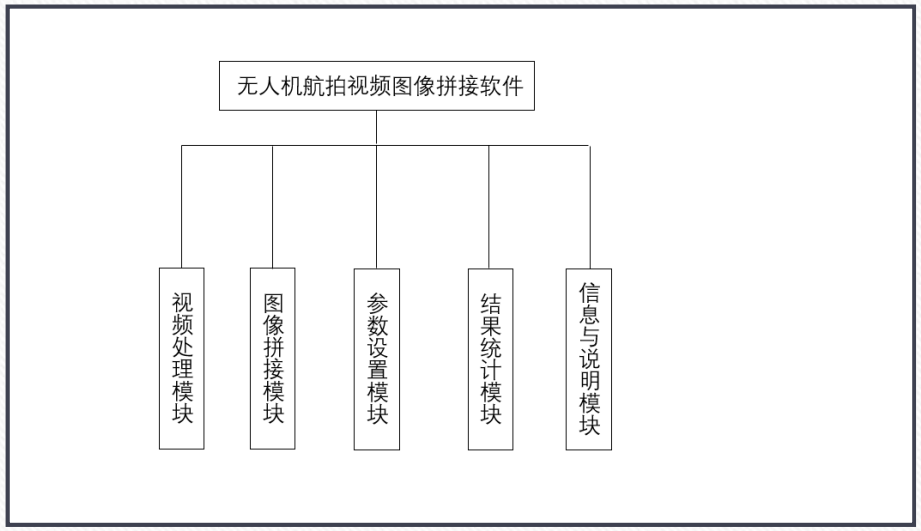
2 对比实验

西安电子科技大学
XIDIAN UNIVERSITY

拼接软件实现与结果对比

04 拼接软件实现与结果对比—拼接软件设计实现

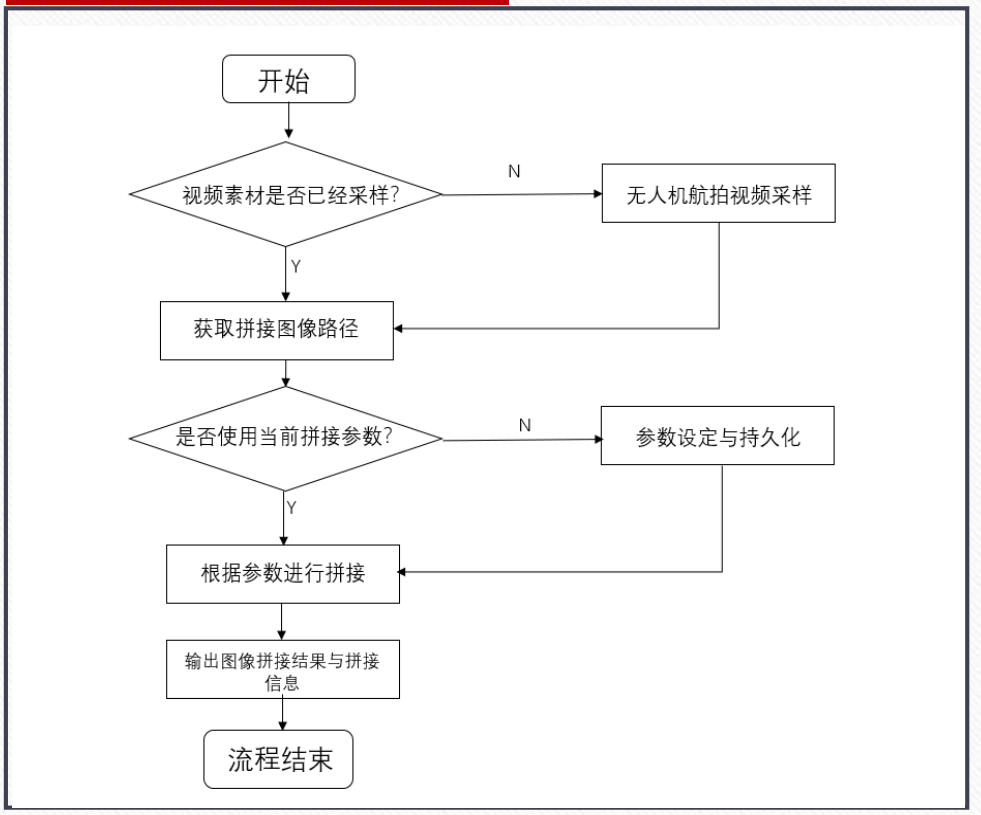
拼接软件模块划分



软件编写环境

算法库：OpenCV 3.4.3+OpenCV Contrib 3.4.3
UI库： QT
编程语言：C++
运行环境：Win10 X64
运行所需空间：1.78G（包含OpenCV库和Cuda库）

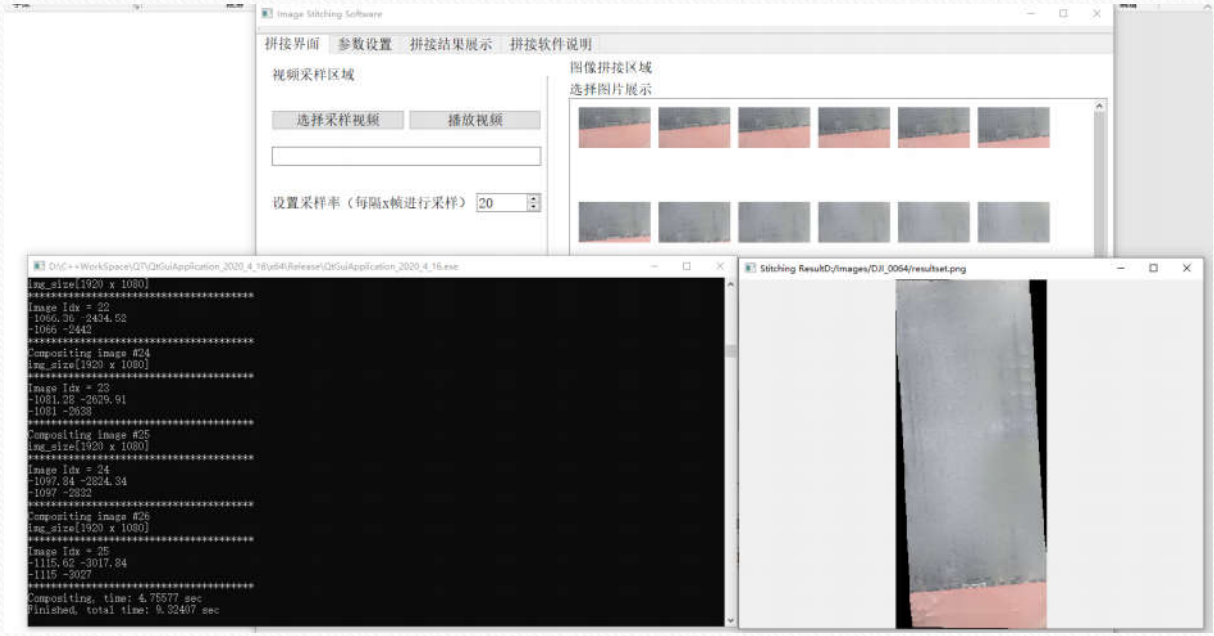
拼接软件处理流程



04 拼接软件实现与结果对比—拼接软件设计实现

视频采样与图像拼接界面

拼接图像效果演示



04 拼接软件实现与结果对比—拼接软件设计实现

参数设置

配置文件截图

Config.ini - 记事本

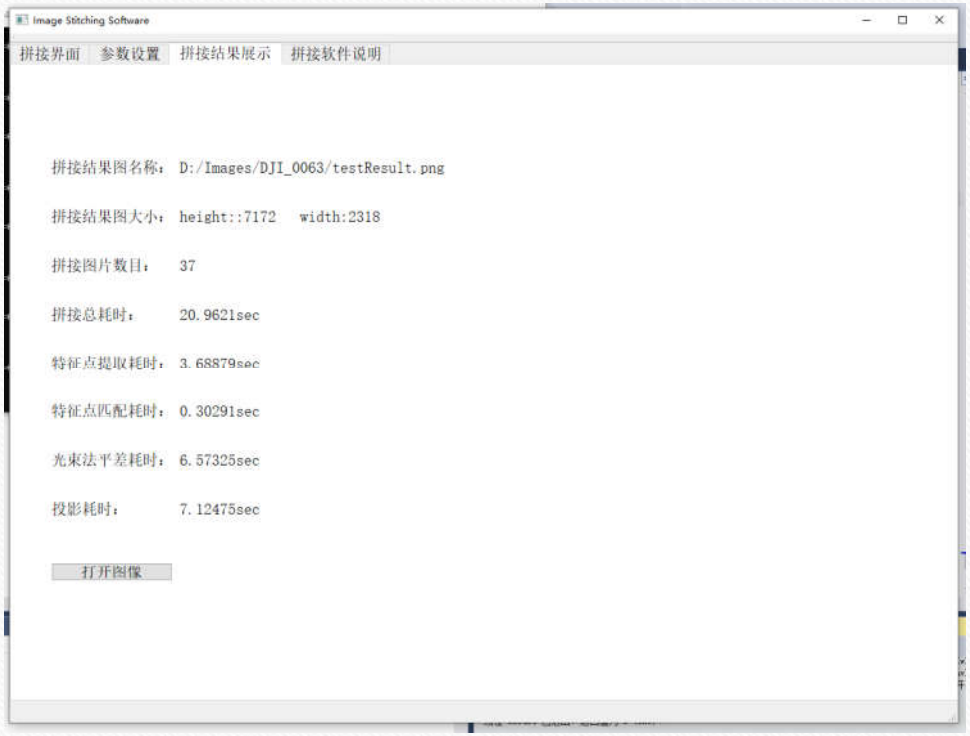
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[multi_processes]
threadNumber=8
usingOpenMP=YES

[properties]
ba_cost_func=ray
blend_type=MULTI_BAND
expos_comp_type=no
features_type=sift
seam_find_type=dp_color
warp_type=plane
```

04 拼接软件实现与结果对比—拼接软件设计实现

拼接结果展示



拼接软件说明

04 拼接软件实现与结果对比—对比实验

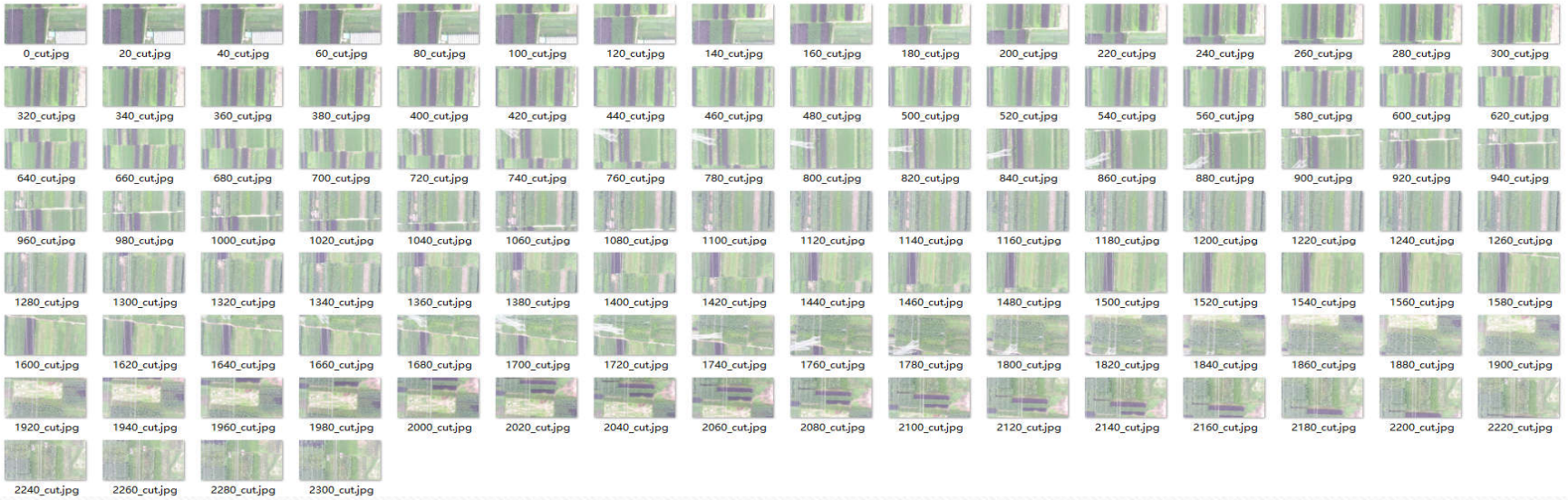
对比软件

PanoramaStudio3Pro.exe

HA.PTGui.Pro.9.1.9.x64.retail.Canniness.tt7z.com.exe

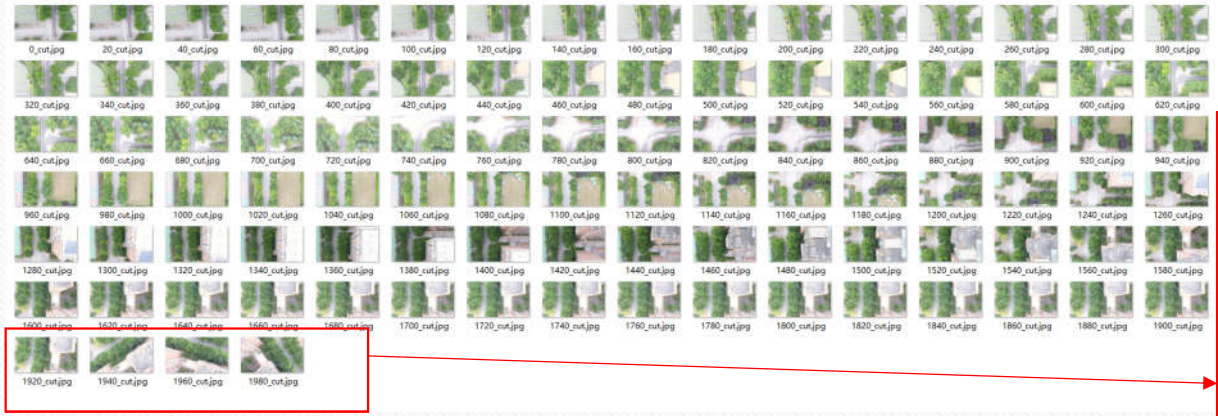
AutoStitch.exe

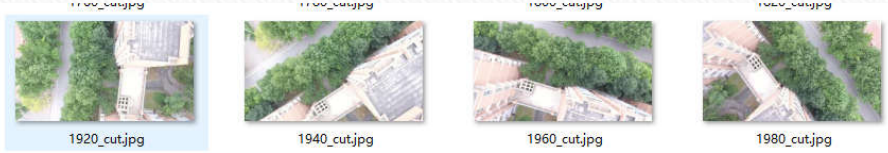
第一组待拼接图像缩略图（116张）



04 拼接软件实现与结果对比—对比实验

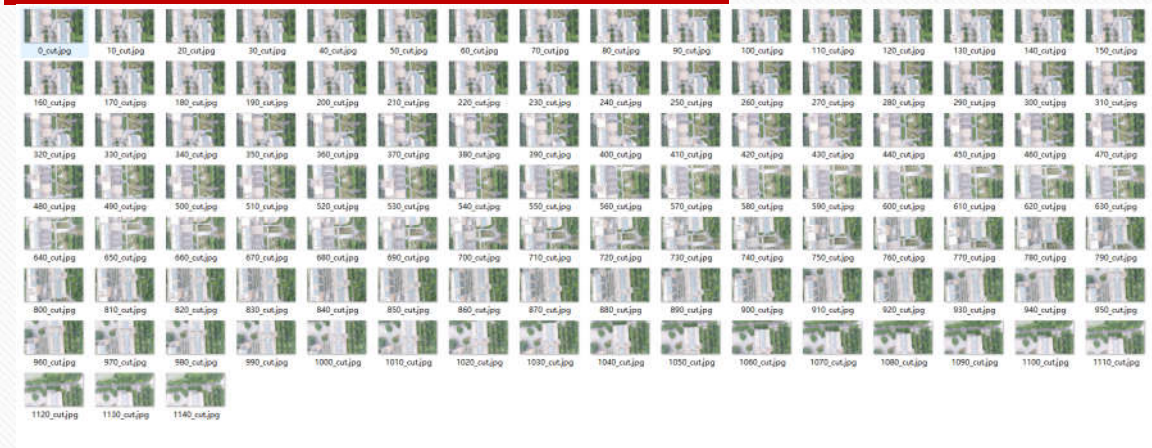
第二组待拼接图像缩略图（100张）





无人机发生偏转（返航）

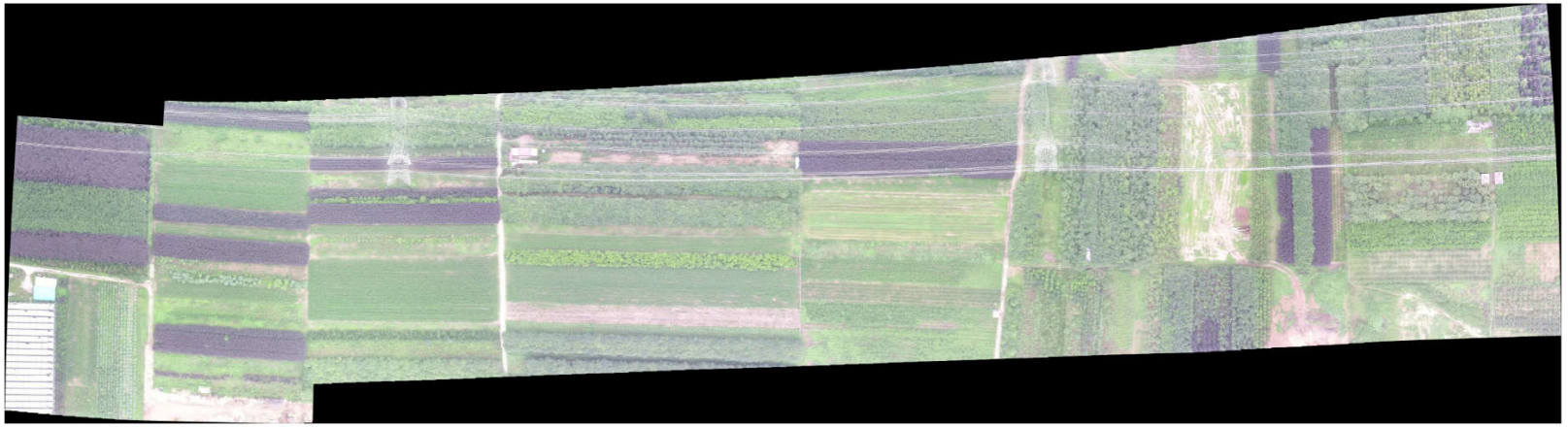
第三组待拼接图像缩略图（115张）



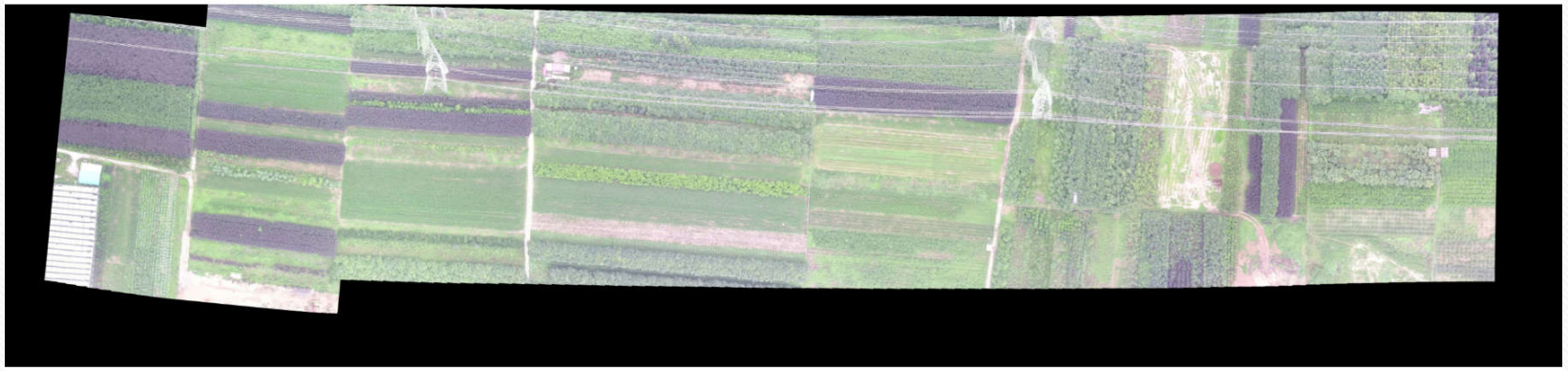
04 拼接软件实现与结果对比—对比实验

第一组拼接结果对比

AutoStitch软件



PTGui软件

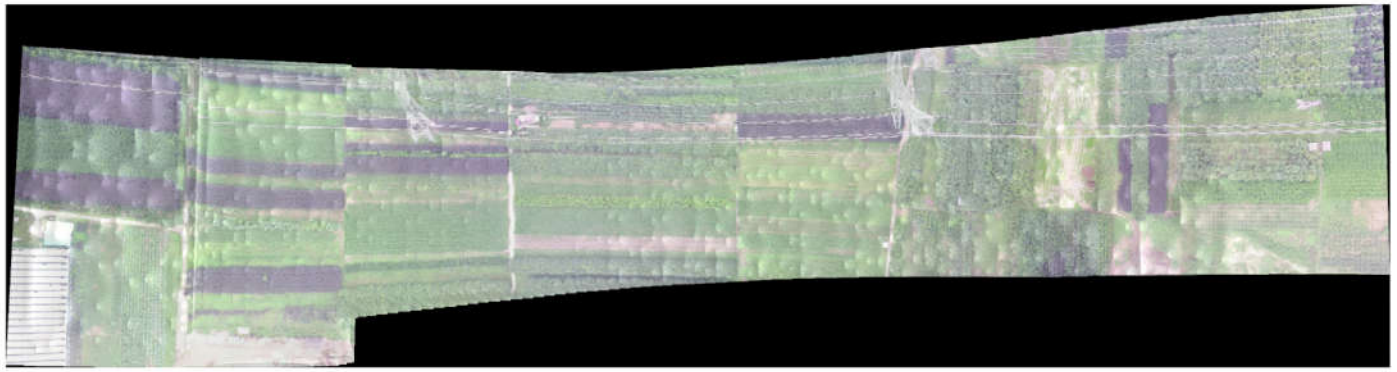


04 拼接软件实现与结果对比—对比实验

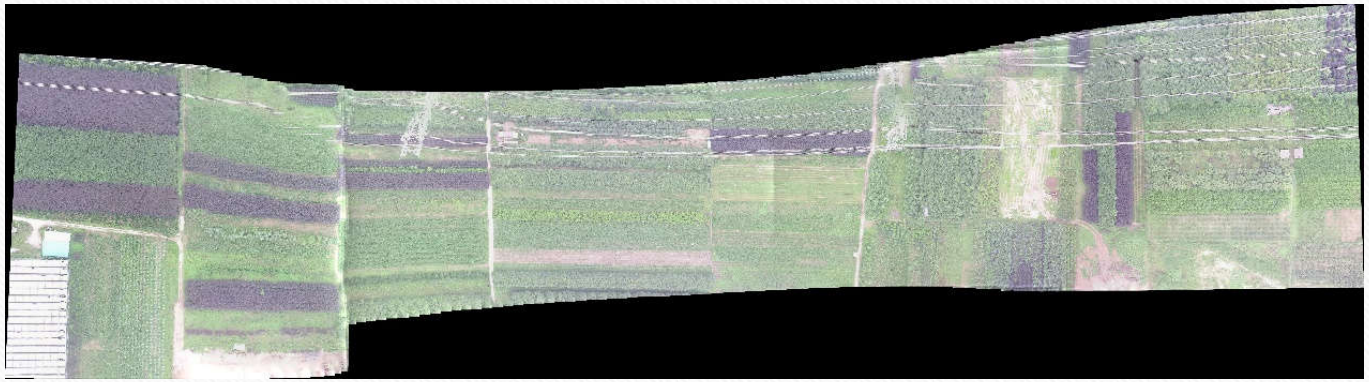
第一组拼接结果对比

Pararama

实现软件（Voronoi）



实现软件（动态规划）



04 拼接软件实现与结果对比—对比实验

第二组拼接结果对比

PTGui软件

Parorama

04 拼接软件实现与结果对比—对比实验

第二组拼接结果对比

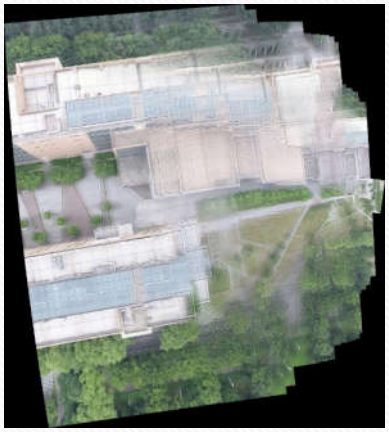
实现软件（Voronoi）

实现软件（动态规划）

04 拼接软件实现与结果对比—对比实验

第三组拼接结果对比

AutoStitch软件



PTGui软件

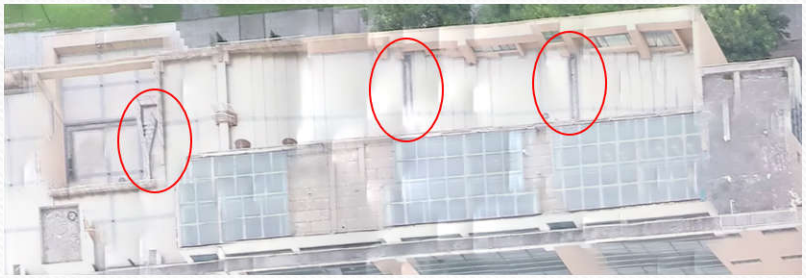
Parorama

04 拼接软件实现与结果对比—对比实验

第三组拼接结果对比

第三组拼接结果细节

实现软件（Voronoi）



实现软件（动态规划）

04 拼接软件实现与结果对比—对比实验

三组拼接耗时对比

拼接软件	第一组（116张）	第二组（100张）	第三组（115张）
AutoStitch 2.2	约174sec	无法拼接	约115 sec(拼接错误)
PTGui	约64sec	约42sec	约62s
本文软件（Voronoi）	109.36 sec	92.6754 sec	70.6461 sec
本文软件（DP）	117.497sec	96.2089 sec	94.4337 sec
PanoramaStudio	约70s(拼接错误)	约60s(拼接错误)	约70s(拼接错误)

总结

1.拼接效果好于AutoStitch2.2和Panorama，还原航拍高分辨率场景;

2.拼接效果上动态规划找接缝线好于Voronoi找接缝线，动态规划法耗时略多，细节上有时候Voronoi更好;

3. 拼接速度优于AutoStitch2.2，不及PTGui。

西安电子科技大学
XIDIAN UNIVERSITY

二次拼接研究

1 单向航拍缺陷

2 方法与流程

3 对比实验

05 二次拼接研究—单向航拍缺陷

单向航拍缺陷

1.只能应用于一维拼接，拍摄中只能在一个方向扩展拼接范围；

2.无人机的通信距离有限，单次遥控距离较短，单向单次飞行不好控制；

3.单次图像拼接的图像过多会导致更严重的耗时问题；

4.民用无人机限高（120m），较低飞行高度难以大范围航拍。

某无人机说明书

安全提示

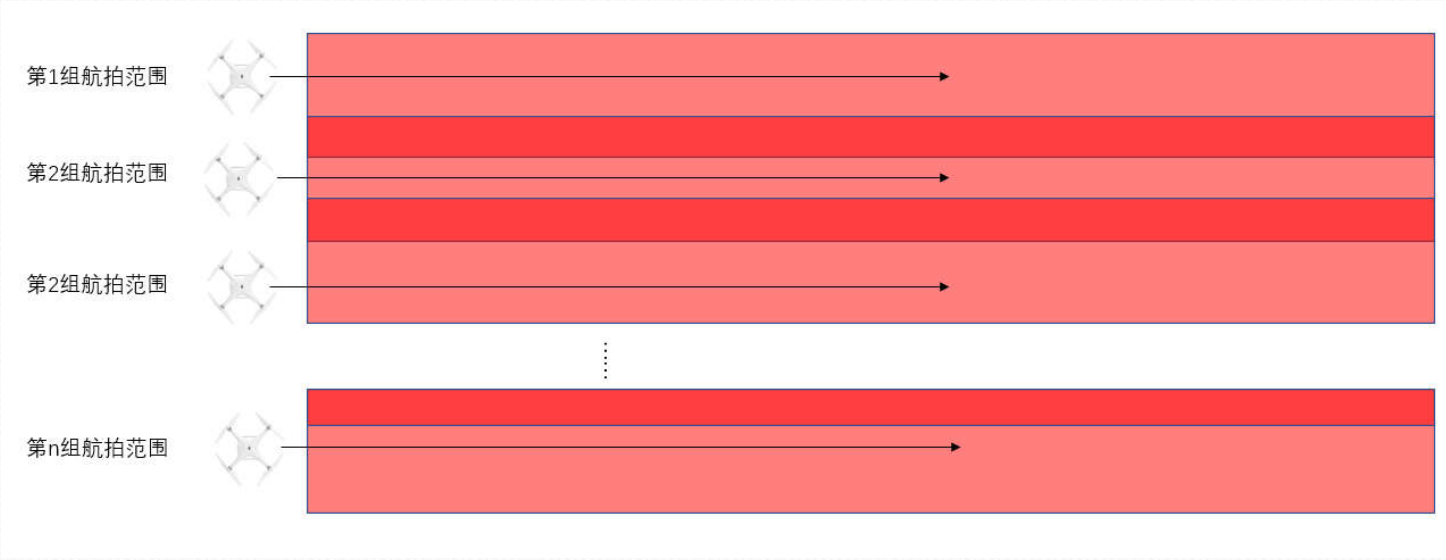
感谢您选择使用飞拍无人机，本手册将引导您快速学习配置和使用您的无人机，为了保障您的飞行安全，请仔细阅读后再进行实际操作，并妥善保管本手册。

- 本产品起飞重量为670g-820g，根据民航局相关政策，起飞重量250g以上的无人机，需要进行无人机实名注册登记，请根据*《无人机实名登记指引卡》进行无人机实名登记。不进行实名登记的，将会被视为非法飞行。
- 根据中华人民共和国相关政策，本无人机飞行高度不得超过起飞点120米。
- 请在起飞前进行安全检查，确保各设备未损坏，螺旋桨无破损、电机无异物，电量充足可正常启动，指南针校准完毕，GPS信号、遥控信号、图传信号良好，各部分均无异常。
- 室内飞行极其危险，请勿在室内飞行。请于开阔无遮挡的[illegible]中飞行，尽量远离建筑物、树木、人群、水面等，确保无人机的飞行不受干扰，安全飞行。
- 请避免于如大风、暴雨等恶劣天气环境中飞行，[illegible]天气环境将可能影响飞行器的飞行稳定性、损坏飞行器性能导致坠毁。
- 请远离通讯基站、大功率天线等电磁信号[illegible]杂地区，避免信号传输干扰导致飞行器失控。
- 请时刻保持飞行器在您自己的视线范围内飞行，不要于可视范围外盲飞，以避免因无法正确判断飞行器姿态而误操作导致飞行器碰撞坠毁。
- 本产品不具备夜间夜视功能，请尽量避免于夜间或图传显示过暗的区域飞行，如遇天色暗淡或天气突变、图传显示过暗或看不清，请立即返航以避免意外。
- 本产品不具备特技飞行功能，请保证每次飞行都“缓、稳、准”，谨慎飞行不心急，操作准确不慌乱，以保证飞行安全。
- 严禁在机场区域、军事区域、敏感机构等敏感禁飞区域周边使用本产品，违者将有可能承担法律风险。
- 请严格遵守当地法律法规，确保飞行安全。

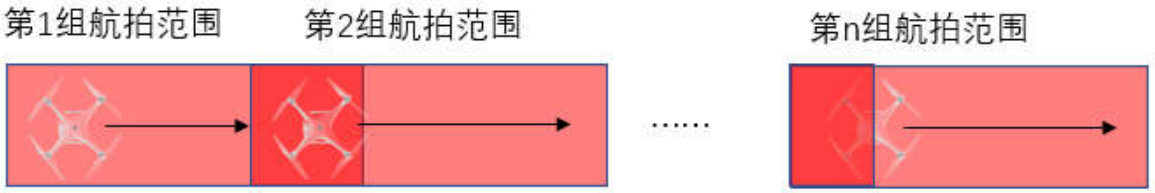
05 二次拼接研究—方法与流程

航拍扩展方法

二维扩展航拍

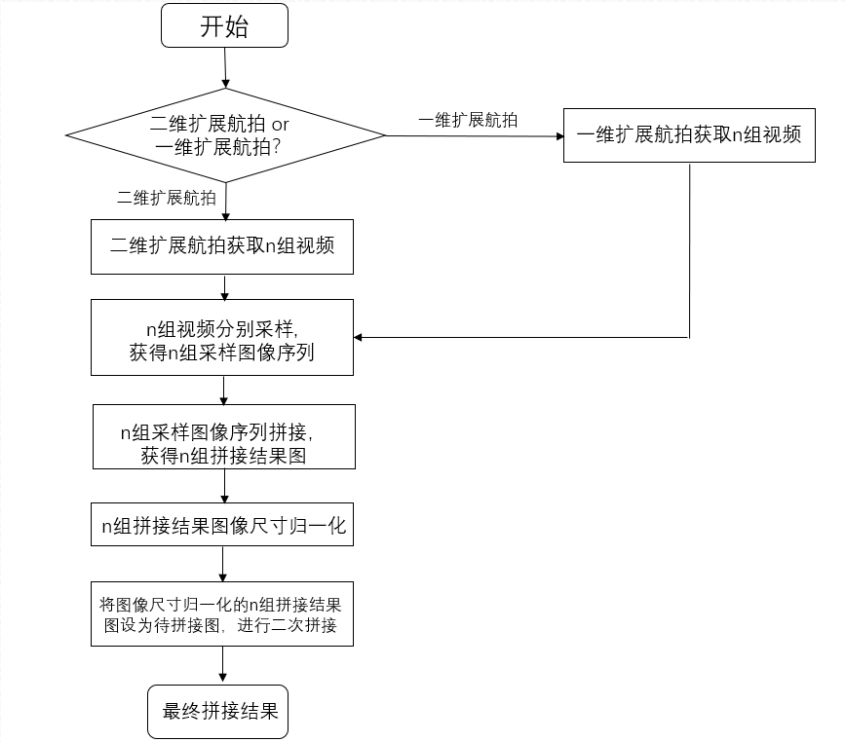


一维扩展航拍



05 二次拼接研究—方法与流程

航拍扩展流程图



航拍扩展实验

1.二维扩展航拍拼接实验：拍摄西电北校区操场，3次单向飞行;

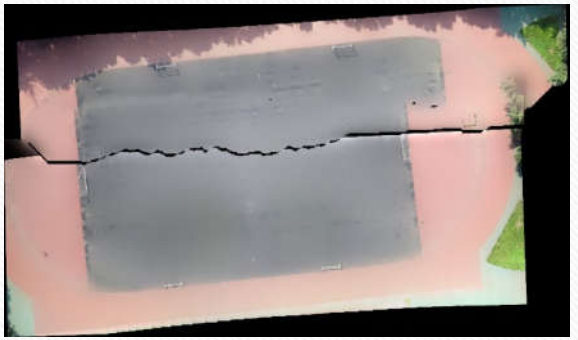
2. 一维扩展航拍拼接实验：拍摄西电南校区教学楼，2次单向飞行。

05 二次拼接研究—对比实验

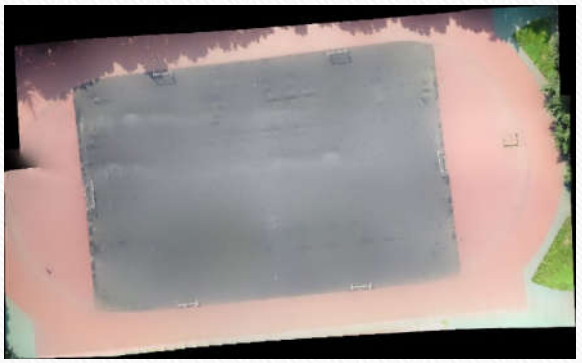
二维扩展航拍（三次单向飞行航拍结果）

拼接对比

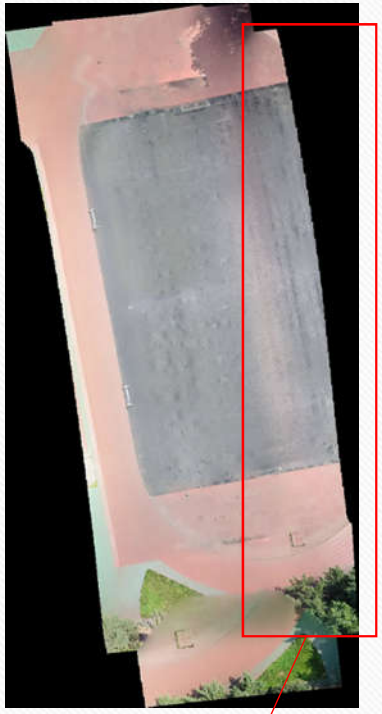
直接拼接



图像尺寸归一化



PTGUI拼接（原始图直接拼接）



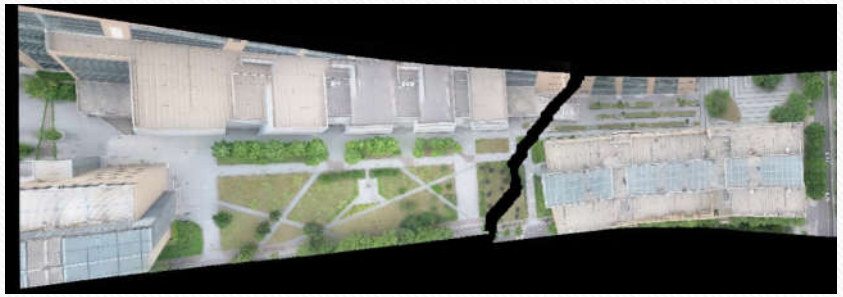
缺失信息

05 二次拼接研究—对比实验

一维扩展航拍（两次单向飞行航拍结果）

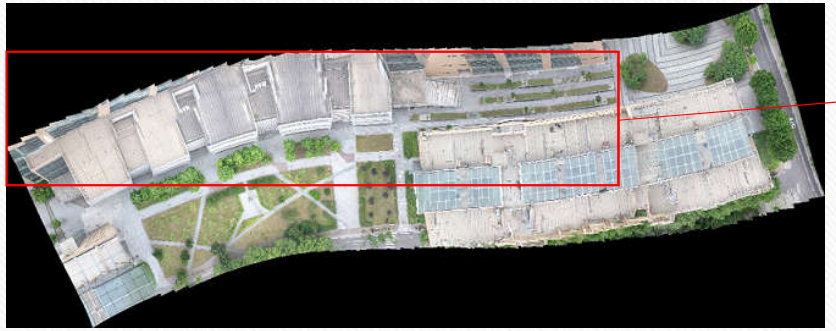
拼接对比

直接拼接



图像尺寸归一化

PTGUI拼接（原始图直接拼接）



建筑弧度不符合现实场景

西安电子科技大学
XIDIAN UNIVERSITY

总结与展望

05 总结与展望—总结

基于阈值改进的光束法平差

统计最近五次LM算法迭代的射线误差

计算方差

低于阈值则跳出

基于降采样的改进拼接流程

对大于10^6像素的图像降采样

融合阶段恢复图像尺寸与其他参数

基于OpenMP多线程加速算法

特征点提取多线程加速

多层融合多线程加速

二次拼接研究

二维扩展航拍拼接

一维扩展航拍拼接

基于改进算法的软件实现

视频采样与图像拼接

参数设置

图像拼接信息统计

05 总结与展望—展望

GPU加速

整合CUDA，
GPU大矩阵乘法，
GPU特征点提取。

矩阵乘法优化

针对雅克比矩阵是稀疏矩阵的特性，使用舒尔补等进行优化。

更多参变量

使用6变量或8变量模型，
改善算法的鲁棒性。

感谢聆听