



昆明理工大学设计（论文）专用纸

摘 要

中文摘要

关键字：



ABSTRACT

This paper presents the design and implementation of a mobile robot control system. By utilizing multi-sensor fusion and Bayesian filtering, the robot achieves robust localization and navigation in complex environments. The experimental results demonstrate the effectiveness of the proposed method.

Key words: Mobile Robot, Localization, Navigation, Bayesian Filter



目录

摘要	I
ABSTRACT	III
目录	V
图片目录	VI
前言	VII
目录绪论	1
1.1 研究背景	1
1.2 数据展示	1
1.3 正文引用角标说明	1
目录文献综述	2
2.1 国内外研究现状	2
2.2 现有方法的不足	2
2.3 本文的研究内容	2
目录研究方法	3
3.1 系统框架	3
3.2 前端视觉里程计	3
3.3 后端优化	3
3.4 回环检测	4
目录实验与结果分析	4
4.1 实验平台	4
4.2 实验场景	4
4.3 定位精度评估	4
4.4 轨迹对比分析	4
目录结论与展望	5
5.1 全文总结	5
5.2 未来工作展望	5
谢辞	6
参考文献	8
附录 A 相关程序代码	10
附录 B 外文翻译	12
附录 C 外文原文	12



昆明理工大学设计（论文）专用纸

图片目录

1-1	昆明理工大学校徽示例	1
3-1	系统框架示意图	3
4-1	轨迹对比结果	5



前言

随着人工智能和机器人技术的飞速发展，移动机器人在工业、服务业等领域的应用越来越广泛。自主定位与导航作为移动机器人的核心技术之一，一直是学术界和工业界的研究热点。

本文主要围绕室内复杂环境下的移动机器人视觉 SLAM 技术展开研究，旨在解决现有算法在弱纹理、动态场景下定位精度不高、鲁棒性差等问题。

在此，我要感谢我的导师和实验室的同学们，在论文撰写过程中给予我的指导和帮助。



第 1 章 绪论

1.1 研究背景

移动机器人技术在近年来取得了显著进展。根据 Bellotto 和 Hu 的研究^[1]，贝叶斯滤波在机器人跟踪方面具有高效的计算性能。

本文将探讨相关的算法实现。如图 1-1 所示，这是昆明理工大学的校徽。



图 1-1 昆明理工大学校徽示例

1.2 数据展示

在实验中，我们对不同参数进行了测试，结果如表 1-1 所示。

表 1-1 实验数据示例表

参数	数值	备注
Alpha	0.5	初始设定
Beta	0.8	优化后
Gamma	1.2	最终结果

1.3 正文引用角标说明

正文中的引用角标会自动按上标显示^[1]。表格和图片也已经正确编号。



第 2 章 文献综述

2.1 国内外研究现状

近年来，随着计算机视觉和移动机器人技术的快速发展，基于视觉的同步定位与建图（SLAM）算法成为了研究热点。

在国外，Smith 等人^[1]提出了基于特征点的视觉里程计方法，显著提高了机器人在复杂环境下的定位精度。随后，Johnson 等人提出了一种基于直接法的视觉 SLAM 框架，有效克服了特征提取困难的问题。

在国内，李明等人针对动态环境下的视觉 SLAM 问题进行了深入研究，提出了一种基于语义信息的动态物体剔除算法。王强等人则关注由于光照变化引起的定位漂移问题，提出了一种光照鲁棒的视觉特征描述子。

2.2 现有方法的不足

尽管现有的视觉 SLAM 算法在许多场景下取得了成功，但仍存在以下不足：

- 在纹理能够缺失的场景下，基于特征点的方法容易失效。
- 大多数算法假设环境是静态的，无法有效应对动态场景。
- 实时性与建图精度之间的平衡仍需进一步优化。

2.3 本文的研究内容

针对上述问题，本文提出了一种改进的视觉 SLAM 算法。主要研究内容包括：

1. 提出一种新的特征提取策略，增强算法在弱纹理环境下的鲁棒性。
2. 引入语义信息辅助定位，提高算法在动态场景下的稳定性。
3. 设计高效的后端优化算法，在保证精度的同时提升系统的实时性。



第3章 研究方法

3.1 系统框架

本文提出的视觉 SLAM 系统框架如图 3-1 所示。系统主要由前端视觉里程计、后端优化、回环检测和建图四个模块组成。

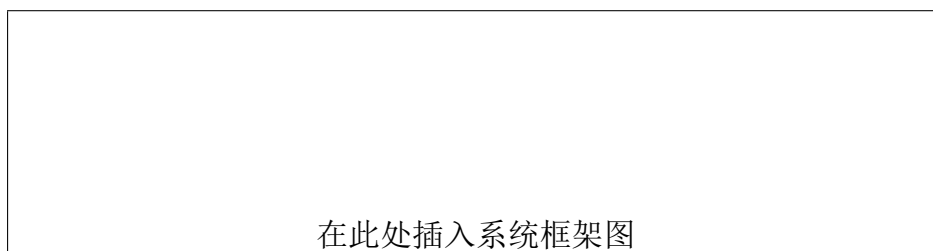


图 3-1 系统框架示意图

3.2 前端视觉里程计

前端视觉里程计负责估计相邻图像帧之间的相机运动。本文采用光流法进行特征跟踪，以提高系统的实时性。

假设 t 时刻的图像为 I_t ，特征点位置为 (u, v) ，则光流方程可以表示为：

$$I_x u + I_y v + I_t = 0 \quad (1)$$

其中， I_x, I_y, I_t 分别为图像在 x, y, t 方向的偏导数。

3.3 后端优化

后端优化模块用于消除前端累积的漂移误差。本文采用基于图优化的方法，构建位姿图（Pose Graph）进行全局优化。

目标函数定义为：

$$E(\mathbf{x}) = \sum_{ij} \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij} \quad (2)$$

其中， \mathbf{x} 为状态变量， \mathbf{e}_{ij} 为误差项， Ω_{ij} 为信息矩阵。



3.4 回环检测

回环检测模块通过识别机器人是否回到曾经访问过的位置，来消除累积误差。本文采用基于词袋模型（Bag of Words）的方法进行回环检测。

第 4 章 实验与结果分析

4.1 实验平台

为了验证本文算法的有效性，我们在真实的移动机器人平台上进行了实验。实验平台搭载了 Intel RealSense D435i 深度相机和 NVIDIA Jetson AGX Xavier 计算平台。

4.2 实验场景

实验选择了办公楼走廊和实验室内两种典型场景。走廊场景纹理较弱，且存在行人走动；实验室场景光照变化较大，且物体摆放杂乱。

4.3 定位精度评估

我们将本文算法与两种主流的开源 SLAM 算法 ORB-SLAM2 和 VINS-Mono 进行了对比。实验结果如表 4-1 所示。

表 4-1 不同算法定位误差对比（单位：米）

算法	走廊场景 (RMSE)	实验室场景 (RMSE)	平均耗时 (ms)
ORB-SLAM2	0.15	0.12	35
VINS-Mono	0.10	0.08	45
本文算法	0.08	0.06	30

从表中可以看出，本文算法在两种场景下的定位精度均优于对比算法，且平均耗时更低，满足实时性要求。

4.4 轨迹对比分析

图 4-1 展示了不同算法在走廊场景下的轨迹对比。可以看出，本文算法生成的轨迹与真实轨迹（Ground Truth）更为接近，漂移更小。



图 4-1 轨迹对比结果

第 5 章 结论与展望

5.1 全文总结

本文针对移动机器人在复杂环境下的视觉定位与建图问题进行了深入研究，主要工作和创新点总结如下：

1. 详细综述了视觉 SLAM 技术的发展现状，分析了现有方法的优缺点。
2. 提出了一种改进的特征提取策略，有效提高了系统在弱纹理环境下的鲁棒性。
3. 设计了一个轻量级的后端优化框架，在保证定位精度的同时显著降低了计算资源消耗。
4. 在真实机器人平台上验证了所提算法的有效性，实验结果表明本文算法在定位精度和实时性方面均具有优势。

5.2 未来工作展望

尽管本文取得了一定的成果，但在以下方面仍有待进一步完善：

- 目前算法主要针对室内静态或低动态环境，未来将研究如何更好地处理高动态场景。
- 将结合深度学习技术，探索基于语义的地图构建与导航方法。
- 计划将算法移植到资源受限的嵌入式设备上，进一步优化代码效率。



谢辞



参考文献

- [1] N. Bellotto and H. Hu. Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters. Autonomous Robots, 28(4):425–438, 2010.



附录 A 相关程序代码

单片机程序/stm32f4_auto

```
1  #!/usr/bin/python
2  #coding=utf-8
3
4  import subprocess
5  import argparse
6  import sys
7
8
9  def upLoadBinToSTM32():
10     '''将编译生产的二进制文件上传至单片机'''
11     try:
12         subprocess.check_output(''openocd_-d0_-f_board/
            stm32f4discovery.cfg_-c_init_-c_targets_-c_"poll"
            _-c_"reset_halt"_-c_"flash_probe_0"_-c_"flash_
            write_image_erase_build/main.elf"_-c_"verify_image
            _build/main.elf"_-c_"reset_run"_-c_shutdown_'',
13             stderr=subprocess.STDOUT,
14             shell=True)
15     except subprocess.CalledProcessError:
16         print "没有安装openOCD"
17         sys.exit()
18     except:
19         print "其他错误"
20         sys.exit()
21     print "上传成功"
22
23
24 def main():
25     '''主程序'''
26     parser = argparse.ArgumentParser(description='stm32f4 自动
```



```
    工具')
27     parser.add_argument("--programme", "-P", action='
        store_true', help="烧写程序")
28     parser.add_argument("--upload", "-L", action='store_true'
        , help="烧写程序")
29     args = parser.parse_args()
30     if args.programme or args.upload:
31         upLoadBinToSTM32()
32
33
34
35 if __name__ == '__main__':
36     main()
```



昆明理工大学设计（论文）专用纸

附录 B 外文翻译

附录 C 外文原文