基于 Raspberry Pi 的 LiDAR 传感器集成与应用开发

黄京

Apr 19, 2025

激光雷达（LiDAR）技术凭借其高精度三维感知能力，已成为自动驾驶、机器人导航和工业检测等领域的核心技术。将低成本 LiDAR 传感器与 Raspberry Pi 结合，不仅为教育科研提供了经济高效的实验平台，更为原型开发打开了创新空间。这种组合使得开发者能够以低于 1000 元人民币的成本构建完整的感知系统，其意义堪比当年 Arduino 对嵌入式开发的革命性影响。

1　LiDAR 基础与硬件选型

激光雷达通过发射激光脉冲并测量反射信号的飞行时间（Time of Flight, ToF）实现测距，其测距公式可表示为：

$$d = \frac{c \cdot \Delta t}{2}$$

其中 c 为光速，Δt 为发射与接收时间差。对于 RPLIDAR A1 这类二维扫描雷达，其水平视场角可达 360°，角分辨率 0.9°，最大测距 12 米，采样频率 8000 点/秒，这些参数使其成为 Raspberry Pi 4B 的理想搭档。
在选择接口方案时，USB 版本 LiDAR 可直接连接 Pi 的 USB 2.0 接口，而 UART 型号需通过 CH340 等转换芯片实现 TTL 电平匹配。需特别注意供电稳定性 —— YDLIDAR X4 要求 5V/500mA 独立供电，若直接使用 Pi 的 GPIO 供电可能导致系统崩溃。

2　硬件集成与驱动配置

以 RPLIDAR A1 为例，硬件连接需遵循「电源隔离」原则：使用外部 5V 电源适配器为 LiDAR 供电，同时通过 USB-TTL 转换器建立数据通道。在 Raspbian 系统上配置时，需修改 `/boot/config.txt` 禁用蓝牙以释放 UART 资源，添加 `enable_uart=1` 配置项并重启。
驱动安装可通过 Python 的 `rplidar-roboticia` 库实现，以下代码展示了基本数据读取逻辑：

```
from rplidar import RPLidar
lidar = RPLidar('/dev/ttyUSB0')
for scan in lidar.iter_scans():
   for (_, angle, distance) in scan:
      if distance > 0:
         radians = angle * (3.1416/180)
         x = distance * math.cos(radians)
         y = distance * math.sin(radians)
```

```
        print(f"坐标 : ({x:.2f}, {y:.2f}) mm")
```

代码中 iter_scans() 方法以生成器形式持续输出扫描数据，每个数据点包含质量、角度和距离三个参数。角度值需转换为弧度制后方可进行直角坐标系换算，math 模块的三角函数实现了极坐标到笛卡尔坐标的转换。实际部署时应添加异常处理机制，防止 USB 端口意外断开导致程序崩溃。

3 应用开发案例

在二维环境地图构建场景中，Hector SLAM 算法因其无需里程计的特性广受青睐。通过 ROS melodic 的 hector_mapping 包，可将 LiDAR 数据实时转换为栅格地图。核心算法通过扫描匹配优化位姿估计，其目标函数可表示为：

$$\Psi^* = \arg\min_{\Psi} \sum_{i=1}^{n} [1 - M(S_i(\Psi))]^2$$

其中 M 为当前地图模型，S_i 表示第 i 个扫描点的坐标变换。

避障机器人开发需关注实时性优化。以下代码片段展示了基于滑动窗口的障碍物检测方法：

```
from collections import deque
class ObstacleDetector:
    def __init__(self, window_size=5):
        self.buffer = deque(maxlen=window_size)

    def update(self, scan_data):
        self.buffer.append(scan_data)
        current_frame = np.mean(self.buffer, axis=0)
        obstacles = current_frame[current_frame[:,2] < 500] # 检测 500mm 内障碍物
        if len(obstacles) > 10: # 超过 10 个点判定为有效障碍
            return self.calculate_escape_angle(obstacles)
        return None

    def calculate_escape_angle(self, points):
        angles = points[:,1]
        hist, bins = np.histogram(angles, bins=36, range=(0, 360))
        safe_sector = np.argmin(hist) * 10 # 10 度分辨率
        return safe_sector + 5 # 返回安全区域中心角度
```

该算法采用 5 帧滑动窗口平滑数据，通过直方图统计寻找障碍物稀疏区域。np.mean 对多帧数据做均值滤波，有效抑制单次扫描噪点。安全角度计算以 10° 为分辨率将周角划分为 36 个扇区，选择点数最少的区域作为逃生方向。

4 优化与挑战

Raspberry Pi 4B 的 CPU 在运行 SLAM 时负载常达 90% 以上，可通过设置 CPU 亲和性优化任务调度。使用 `taskset` 命令将关键进程绑定至特定核心：

```
taskset -c 3 roslaunch hector_slam tutorial.launch
```

此举将 SLAM 算法限制在第三个 CPU 核心运行，避免任务切换开销。数据降噪方面，Savitzky-Golay 滤波器在保留特征的同时有效平滑轨迹，其卷积核函数为：

$$y_i = \sum_{j=-m}^{m} c_j x_{i+j}$$

其中 c_j 为最小二乘拟合系数，窗口大小 m 一般取 5-11 的奇数值。实测表明，当扫描频率为 5Hz 时，7 点窗口可使均方误差降低 62%。

5 未来趋势与资源推荐

固态 LiDAR 技术正突破传统机械式结构的成本瓶颈，美国 Velodyne 最新发布的 Velarray H800 已实现 200 米探测距离与 120° 视场角，而价格仅为前代产品的三分之一。学习路径建议从 ROS 官方文档入手，结合《Probabilistic Robotics》理论专著，再通过 GitHub 上的 `rplidar_ros` 等开源项目实践提升。

当开发者成功实现首个 LiDAR 应用时，那种透过代码「看见」物理世界的震撼，正是技术创新最纯粹的乐趣。期待每位读者都能在这个融合光电子与嵌入式开发的领域，找到属于自己的突破点。