自動送餐機器人(模擬)

學生姓名：戴宇珓、陳信良　　指導老師：林柏江

1. 前言

隨著科技的快速發展，機器人技術在各個領域的應用越來越廣泛。其中，送餐機器人作為一種智能化服務設備，正在改變餐飲作業的傳統服務模式。送餐機器人不僅能提高餐廳的服務效率，減少人力成本，還能在疫情等特殊情況下降低人員接觸風險，保障用戶和員工的安全。因此，研究和應用送餐機器人具有重要的實際意義和廣闊的市場前景。

近年來，許多企業和科研機構紛紛投入到送餐機器人的研發中，並取得了顯著的進展。然而，現有的送餐機器人仍然面臨諸多挑戰，例如自主導航，障礙物檢測與避讓、穩定性以及人機交互等問題。因此，進一步研究和改進送餐機器人的技術，以提高性能和用戶體驗，是一個亟待解決的重要課題。

本文旨在對送餐機器人的關鍵技術和應用現狀進行系統研究，並探討其未來發展。具體來說，我們將重點關注以下幾個方法：送餐機器人的導航和定位技術、環境感知與障礙物避讓技術、機器人的穩定性設計以及人機交互介面的設計。通過綜合分析和實驗驗證，我們期盼能夠為送餐機器人的進一步發展提供有價值的參考和借鑒

2. 主要內容

2.1 理論

2.1.1 ROS (Robot Operating System)

ROS 提供了一個靈活和模塊化的框架，用於構建機器人的導航和控制系統。我們利用 ROS 來處理機器人的感知、計算和行動，通過 ROS 節點進行通信和協調。在 ROS 中，topic、service 和 action 是三種基本的通信機制，用於不同的通信需求和應用場景。

Topic：適用於頻繁的數據流傳輸，非同步和單向通信。

Service：適用於即時響應的請求-響應模式，同步通信。

Action：適用於需要反饋和持續時間較長的任務，有反饋機制和目標取消功能。

而本文設計的機器人自動導航主要應用了 ROS 中的 Action 機制。Action

機制非常適合機器人導航這種需要持續一段時間並且需要反饋的任務。機器人導航包括指定目標位置，並且在導航過程中需要持續監控和反饋導航狀態。

機器人導航系統通過 ROS 的 Action 機制實現，move_base 節點作為 Action Server 處理導航任務，並提供反饋。GUI 和 RobotController 作為 Action Client 發送導航目標並等待結果，實現了自動導航功能。這種架構使得系統具備很好的擴展性和靈活性，適用於各種自動導航應用。

詳細機制如下：

Action Client：

GUI 或者 RobotController 扮演 Action Client 的角色。當用戶在 GUI 上點擊導航按鈕時，會通過 RobotController 發送導航目標。在 robot_controller.py 中，send_goal 方法發送導航目標並等待結果。

Action Server：

move_base 節點充當 Action Server 的角色。它接收導航目標，規劃路徑，並控制機器人移動。move_base 節點還會在導航過程中提供持續的反饋，如當前機器人位置、導航狀態等。

具體流程：

發送導航目標：

當用戶在 GUI 中選擇目標位置（例如 "去座位 1"）時，RobotController 的相應方法（如 go_to_seat1）會被調用。該方法使用 send_goal 函數創建一個新的導航目標，並通過 Action Client 發送給 move_base。

導航過程：

move_base 節點接收到導航目標後，開始進行路徑規劃和移動控制。

在導航過程中，move_base 會持續向 Action Client 提供反饋，如當前的位置、距離目標的距離等。

導航完成：

當機器人到達目標位置（或者接近目標位置，根據我們設置的範圍），move_base 會通知 Action Client 導航完成。send_goal 方法中會等待導航結果，導航完成後返回結果並結束這次導航。

2.1.2 PyQt5 的基本原理

PyQt5 是一個 Python 綁定的 Qt 應用程式架構，它提供了創建跨平台 GUI 應用程式的能力。涵蓋了所有 Qt 的所有功能，包括窗口管理、小部件、事件處理、圖形和多媒體等。

2.1.3 Signal 與 Slot 機制

Signal 與 Slot 機制是 PyQt5 的核心概念之一，用於處裡事件的交互。Signal 是對某個事件的通知，當一個特定事件發生時，該事件會發出一個訊號。例如，按鈕被點擊時會發出 clicked 訊號，文本框中文字改變時會發出 text Changed 訊號。而 Slot 則是對 Signal 的回應函數，負責處理信號發出的事件。Slot 函數可以是任何可調用的 Python 函數，包括內置函數、用戶定義函數等。

當 Signal 發出時，與之連接的 Slot 函數將會被調用，從而實現介面元素之間的互動，例如，按鈕的點擊可以觸發一個 Slot 函數來更新標籤的文本，如圖 1 所示。

```python
from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QLabel, QVBoxLayout, QWidget

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Singal與Slot示例")
        self.layout = QVBoxLayout()
        self.button = QPushButton("點擊我")
        self.layout.addWidget(self.button)
        self.label = QLabel("狀態：未點擊")
        self.layout.addWidget(self.label)
        container = QWidget()
        container.setLayout(self.layout)
        self.setCentralWidget(container)

        # 連接Singal與slot
        self.button.clicked.connect(self.on_button_clicked)

    def on_button_clicked(self):
        self.label.setText("狀態：已點擊")

app = QApplication([])
window = MainWindow()
window.show()
app.exec_()
```

圖 1. Signal 與 Slot 機制範例

同個 Slot 函數可以連接到不同的 Signal 抑或是相反一個 Signal 連接到不同的 Slot，這些所有的 Slot 函數都會在 Signal 發出時被調用。Signal 與 Slot 機制是 PyQt5 強大且靈活的事件處理機制，允許開發者輕鬆地實現對象之間的通訊。通過合理設計和使用 Signal 與 Slot，可以構建高效且響應迅速的 GUI 應用程式。

2.1.4 介面設計

PyQt5 提供了許多小部件和佈局方式，用於構建直觀且功能強大的用戶介面。設計介面時也可以使用 Qt Designer 進行可視化設計，並通過 Python 代碼進行邏輯實現。

2.1.5 SLAM (Simultaneous Localization and Mapping)

自我定位：

SLAM 使機器人能夠在未知或動態變化的環境中確定自身的位置。這對於機器人來說是至關重要的，因為機器人需要知道自己的位置才能進行有效的導航和路徑規劃。

地圖構建：

SLAM 使機器人能夠在移動過程中構建或更新周圍環境的地圖。這些地圖可以用來進行後續的導航、障礙物避讓等操作。

自主導航：

通過 SLAM，機器人可以在未知環境中自主探索，找到最優路徑到達目標位置。SLAM 技術使機器人能夠在不依賴外部基礎設施（如 GPS）的情況下實現精確的導航。

環境感知：

SLAM 技術使機器人能夠感知和記錄周圍的環境特徵，如牆壁、障礙物、門等，這些信息對於機器人的任務執行和環境交互非常重要。

動態更新：

在動態環境中，SLAM 技術使機器人能夠即時更新其地圖和位置，這對於處理環境中出現的動態障礙物（如人、其他機器人）非常關鍵。

2.2 實作

2.2.1 終端啟動 launch 檔案

此檔包含設定 TurtleBot3 model 為 burger 模型、開啟 ROS 的機器人自動導航等多功能(包含 map server、move_base 等功能)以及開啟 RViz 以方便觀察機器人實時定位及導航位置。

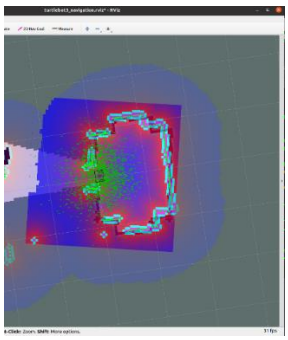

圖 3. RViz

2.2.2 開啟主要 Python 檔案

主要分為三個檔案，分別是與 ROS 結合用來操控機器人實際行動的 controller 檔案、設定 UI 介面的主要檔案以及結合兩者的檔案。

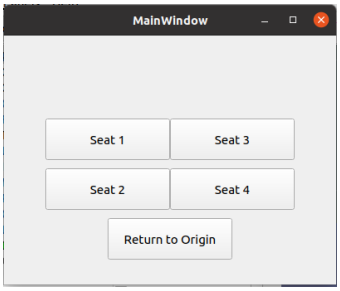

圖 4. UI 操控介面

2.2.3 實際模擬

實際模擬自動送餐機器人時，把餐桌訂在四個角落，分別為 1~4 號桌子，並把入口設為廚房門口，也是機器人的初始位置(在 UI 介面的名稱為 Return to Origin)。

實際模擬時，可以在 UI 介面任意選取 1~4 號桌及初始位置按鈕(廚房)，選取完之後機器人就會立即選擇最短路徑導航到該位置，並且機器人移動的路上會不斷更新回傳自身當前位置，雷達也會不斷掃描周圍障礙物，如果遇到障礙物，機器人雷達掃描之後會立即更新(避開障礙物之後的)最短路徑，此程式會持續到機器人達成命令抵達目的地。

如果 UI 介面點選目的地為 1 號桌，卻在機器人移動的路上突然又點選新命令(例如回到原點或是 2~4 號桌)，那機器人會立即取消上一個命令，執行新命令，並從當前位置執行導航功能到新目的地。

圖 5. 實際模擬畫面

2.3 結論

本文針對 ROS 應用在 turtlebot3 的 burger 模型來模擬送餐機器人以及 UI 人機交互介面設計進行研究，重點探討了如何應用 PyQt5 來實現高效、友好且直觀的用戶介面，通過設計和實驗，我們得出主要幾個觀點：

1. 機器人雷達問題：turtlebot3 的 burger 機器人在實際使用時，雷達掃描不到比機器人低的小台階(障礙物)，因此會造成機器人被卡住，然而透過 RViz 可以觀察到機器人沒掃描到小台階，因此機器人在 map 中會繼續往前，造成實際位置與機器人認定在地圖中的位置產生極大誤差。
2. 導航穩定性：機器人在一開始實驗模擬時，會發生抵達目的地卻始終不停下來，原因為雷達過於敏感，持續掃描到周圍的牆壁或桌子，所以在設定新場地(map)時，需要因應場地設定雷達以及導航到目的地的範圍誤差大小，使機器人抵達目的地之後會快速停下，不會卡住。
3. 介面友好性：基於 PyQt5 開發的送餐機器人介面具有高度的可制定性以及靈活性。在設計層面上 PyQt5 直觀、易用、能夠簡單提升用戶的操作體驗，使用戶能夠輕鬆完成送餐機器人的各種指令和設置。
4. 交互效率：利用 PyQt5 的訊號機制，可以實現較快的事件處理和反應，提高了送餐機器人介面速度，降低了用戶操作等待時間。依我們設計的操作介面仍可以進一步優化。

總結來說，基於 PyQt5 開發的送餐機器人人機交互介面設計和實現上取得進展，我們也認識到介面設計存在一些挑戰，如何進一步提升交互的智能化以及功能的完備性，實現更自然的操作抑或是語音交互；而 ROS 應用在 turtlebot3 的 burger 模型上的實驗結果為非常彈性，可以讓用戶自由發展創意，例如利用 python 設計，但必須注意的就是機器人在每次應用的新場地，都需要客製化調整，如雷達敏感度、地圖障礙物牆壁的高度等，除此之外便可自由發揮。

本文實驗未來的研究可以在以下幾點方面展開，如更易懂操作的圖形介面，增強用戶與機器人的溝通體驗；個性化定制，開發更靈活的介面定制功能，使用戶能夠根據個人偏好和需求調整介面的布局以及功能；跨平台應用，探索介面的跨平台實現，確保在不同操作系統和設備(IOS 或是安卓等)的一致性和流暢性。

3. 參考文獻

[1] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim. (2017). ROS Robot Programming. ROBOTIS Co.,Ltd.