Разработка технической документации для системы навигации автономного робота

Название программы: Система навигации автономного робота

Версия: 1.0

Автор: Повшедный А.Д.

Дата: 01.01.2024

Содержание

Оглавление

Разработка технической документации для системы навигации автономного робота 1
Содержание 2
Введение 3
Описание алгоритмов 3
Общая концепция системы 3
Используемые алгоритмы 3
Алгоритм A* (A-star) 4
Основные шаги алгоритма A*: 4
Применение в системе 5
Руководство по установке и настройке 5
Системные требования 5
Инструкции по установке зависимостей и запуску системы 5
Генерация карты пространства и задание препятствий 6
Пример кода для генерации карты и задания препятствий: 6
Руководство пользователя 7
Использование системы навигации 7
Пример кода для задания начальной и конечной точек и визуализации маршрута: 7
Обработка ошибок 7
Технические детали 8
Основные компоненты 8
Структура навигационного модуля: 8
Используемые библиотеки 8
Структуры данных 8
Примеры и случаи использования 8
Примеры использования системы навигации 9
Пример кода с комментариями: 9
Приложения 10
Приложение А: Схема системы навигации 10
Приложение Б: Дополнительные материалы 10

Введение

В документе описаны основные компоненты системы, используемые алгоритмы и их реализация, а также приведены примеры использования системы в различных условиях.

Описание алгоритмов

Общая концепция системы

Система навигации автономного робота предназначена для обеспечения безопасного и эффективного перемещения робота в пространстве с препятствиями. Она собирает информацию о текущем окружении робота с помощью сенсоров, строит карту пространства и планирует оптимальные маршруты движения, обходя препятствия. Основные задачи системы включают:

- Сбор данных о пространстве и препятствиях.
- Построение карты пространства.
- Планирование маршрута от начальной до конечной точки.
- Обход препятствий и адаптация маршрута в реальном времени.

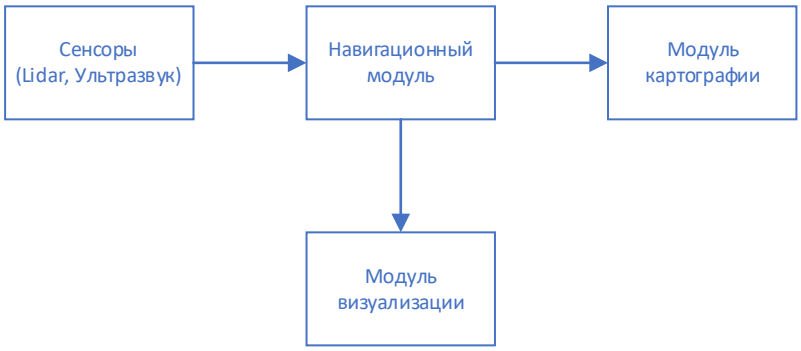

Рисунок 1

Используемые алгоритмы

Основной алгоритм, используемый для поиска кратчайшего пути в системе, — это алгоритм A* (A-star). Этот алгоритм эффективно находит оптимальные маршруты, учитывая препятствия на пути робота. Алгоритм A* объединяет преимущества алгоритмов поиска в ширину и поиска по наилучшей оценке, используя эвристическую функцию для ускорения поиска.

Алгоритм A* (A-star)

Алгоритм A* (A-star) — это метод поиска кратчайшего пути, который использует эвристические оценки для улучшения скорости поиска. Алгоритм комбинирует стоимость пути от начальной точки до текущей (G) и эвристическую оценку оставшейся стоимости пути до конечной точки (H).

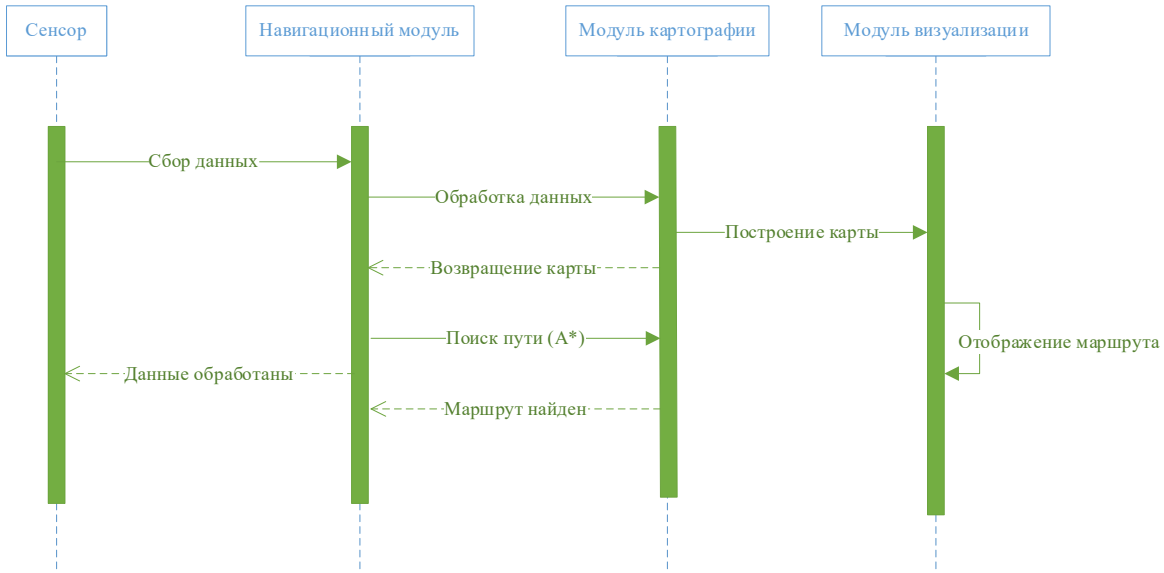

Рисунок 2

Основные шаги алгоритма A*:

1. Инициализация открытого и закрытого списков.
2. Добавление начальной точки в открытый список.
3. Пока открытый список не пуст:
   - Извлечение узла с наименьшей оценкой F = G + H из открытого списка.
   - Если узел является целевой точкой, маршрут найден.
   - Перемещение узла в закрытый список.
   - Для каждого соседнего узла:
     - Если узел уже в закрытом списке, пропустить его.
     - Если узел не в открытом списке или найден более короткий путь к узлу, обновить его данные и добавить в открытый список.
4. Если открытый список пуст, маршрута нет.

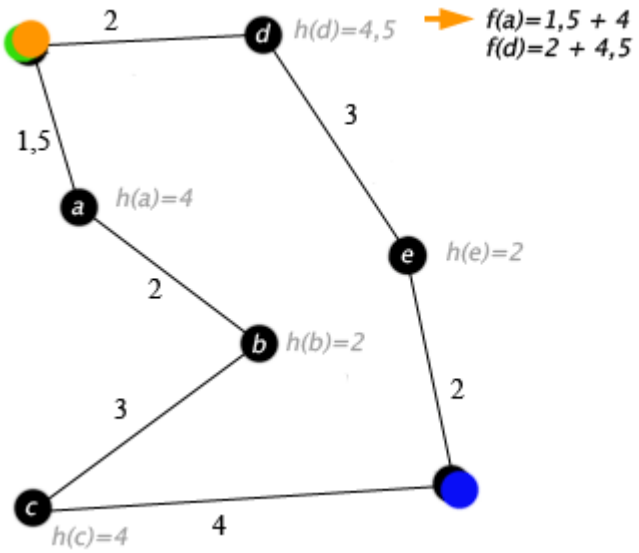

Рисунок 3

Применение в системе

Алгоритм A* используется для расчета пути от начальной до конечной точки на карте пространства робота. Он учитывает текущие координаты робота, целевую точку и расположение препятствий, чтобы построить оптимальный маршрут.

Руководство по установке и настройке

Системные требования

- **Операционная система**: Linux, Windows, MacOS
- **Процессор**: не менее 2.5 ГГц
- **Оперативная память**: не менее 4 ГБ
- **Дисковое пространство**: не менее 500 МБ
- **Дополнительные компоненты**: сенсоры для сбора данных о пространстве (LIDAR, ультразвуковые датчики и т.д.)

Инструкции по установке зависимостей и запуску системы

1. **Установка зависимостей**:

```
Sudo apt-get update
sudo apt-get install python3 python3-pip
pip3 install numpy scipy matplotlib
```

2. **Запуск системы**:

```
python3 navigation_system.py
```

Генерация карты пространства и задание препятствий

1. **Генерация карты**:
   - Используйте встроенные функции для создания карты пространства.
   - Определите размеры карты и начальные параметры.
   - Задайте координаты препятствий вручную или с использованием сенсоров.
2. **Задание препятствий**:

```
map.set_obstacle(x, y)
```

Пример кода для генерации карты и задания препятствий:

```
01: import numpy as np                                  # Импорт библиотеки numpy
02: import matplotlib.pyplot as plt                     # Импорт библиотеки matplotlib для визуализации

03: # Создание карты 20x20
04: map = np.zeros((20, 20))                            # Создание пустой карты размером 20x20

05: # Задание препятствий
06: map[5, 5:15] = 1                                    # Установка горизонтального препятствия
07: map[10:15, 10] = 1                                  # Установка вертикального препятствия

08: # Функция для отображения карты
09: def display_map(map):                               # Определение функции display_map
10:     plt.imshow(map, cmap='gray')                    # Отображение карты в оттенках серого
11:     plt.show()                                      # Показать карту
```

```
12: display_map(map)                                    # Вызов функции
display_map
```

Руководство пользователя

Использование системы навигации

1. **Задание начальной и конечной точек**:

```
navigator.set_start(x_start, y_start)
navigator.set_goal(x_goal, y_goal)
```

2. **Интерпретация результатов поиска пути**:

- После выполнения поиска пути система вернет массив координат, представляющий маршрут.
- Визуализируйте маршрут на карте.

3. **Визуализация карты и пути робота**:

```
navigator.visualize_path()
```

Пример кода для задания начальной и конечной точек и визуализации маршрута:

```
01: # Задание начальной и конечной точек
02: start = (0, 0)                                      # Начальная точка
(0, 0)
03: goal = (10, 10)                                     # Конечная точка
(10, 10)

04: # Запуск поиска пути
05: path = navigator.find_path(start, goal)             # Поиск пути с
помощью навигатора

06: # Визуализация маршрута
07: navigator.visualize_path(map, path)                 # Визуализация пути
на карте
```

Обработка ошибок

Отсутствие пути: если маршрут не найден, убедитесь в корректности задания начальной и конечной точек и проверьте наличие проходимых путей между ними. Возможно, потребуется пересчитать маршрут, учитывая новые данные от сенсоров.

Технические детали

Основные компоненты

- **Навигационный модуль**: управляет процессом поиска пути и взаимодействует с сенсорами.
- **Модуль картографии**: отвечает за создание и обновление карты пространства.
- **Модуль визуализации**: отвечает за отображение карты и маршрутов.

Структура навигационного модуля:

1. **Сбор данных от сенсоров**:
   - Сбор данных о препятствиях и свободных зонах.
2. **Построение карты пространства**:
   - Создание и обновление карты в реальном времени.
3. **Планирование маршрута**:
   - Использование алгоритма A* для поиска кратчайшего пути.

Используемые библиотеки

- **numpy**: для работы с массивами данных.
- **scipy**: для научных вычислений.
- **matplotlib**: для визуализации данных.

Структуры данных

- **Карта пространства**: Двумерный массив, представляющий рабочую область робота.
- **Очередь приоритетов**: используется в алгоритме A* для хранения узлов.

Примеры и случаи использования

Примеры использования системы навигации

1. **Сценарий 1: Робот в офисном помещении**
    1. **Задача**: Доставка предмета от одной комнаты к другой.
    2. **Процесс**: Задание начальной и конечной точек, генерация пути, обход препятствий.
    3. **Пример кода**:

```
01: start = (1, 1)                                  # Начальная точка (1, 1)
02: goal = (15, 15)                                 # Конечная точка (15, 15)
03: path = navigator.find_path(start, goal)         # Поиск пути
04: navigator.visualize_path(map, path)             # Визуализация пути
```

2. **Сценарий 2: Робот на складе**

- **Задача**: Автоматизация складских операций.
- **Процесс**: Задание маршрутов для робота, обновление карты при изменении обстановки.
- **Пример кода**:

```
01: start = (2, 2)                                  # Начальная точка (2, 2)

02: goal = (18, 18)                                 # Конечная точка (18, 18)
03: path = navigator.find_path(start, goal)         # Поиск пути
04: navigator.visualize_path(map, path)             # Визуализация пути
```

Пример кода с комментариями:

```
01: import numpy as np                              # Импорт библиотеки numpy
02: import matplotlib.pyplot as plt                 # Импорт библиотеки matplotlib для визуализации

03: # Задание начальной и конечной точек
```

```
04: start = (0, 0)                                    # Начальная точка (0, 0)
05: goal = (10, 10)                                   # Конечная точка (10, 10)

06: # Инициализация карты
07: map = np.zeros((20, 20))                          # Создание пустой карты размером 20x20

08: # Задание препятствий
09: map[5, 5:15] = 1                                  # Установка горизонтального препятствия

10: # Функция для визуализации
11: def visualize_path(map, path):                    # Определение функции visualize_path
12:     plt.imshow(map, cmap='gray')                  # Отображение карты в оттенках серого
13:     for point in path:                            # Проход по точкам пути
14:         plt.plot(point[1], point[0], 'bo')        # Отображение точки пути
15:     plt.show()                                    # Показать карту

16: # Пример использования алгоритма A*
17: path = a_star_algorithm(map, start, goal)         # Запуск алгоритма A* для поиска пути
18: visualize_path(map, path)                         # Визуализация найденного пути
```

Приложения

Приложение А: Схема системы навигации

Схема архитектуры системы: (См. Рисунок *1*.)

Диаграмма последовательности: (См. Рисунок *2*).

Пример алгоритма: (См. Рисунок *3*).

Приложение Б: Дополнительные материалы

Список используемых библиотек и их версий:

- numpy: версия 1.21.2
- scipy: версия 1.7.1
- matplotlib: версия 3.4.3