Politechnika Warszawska

WYDZIAŁ MECHANICZNY
ENERGETYKI I LOTNICTWA

Generowanie trajektorii 2D robota mobilnego z wykorzystaniem algorytmu Dijkstry

Mikołaj Cebula 304486

WARSZAWA 2024

Spis treści

1. Wstęp

Tematem projektu jest program generujący trajektorię 2D dla robota mobilnego. W tym celu wykorzystano algorytm Dijkstry, który na podstawie grafu znajduje najkrótszą ścieżkę przejazdu z punktu początkowego do końcowego.

2. Założenia projektu

Celem projektu jest zaimplementowanie algorytmu Dijkstry oraz jego przetestowanie. Program ma znajdować najkrótszą ścieżkę przejścia z punktu początkowego do końcowego.

2.1. Struktura programu

W pierwszej części programu generowana jest macierz mająca na celu odzwierciedlenie obszaru pracy robota. Realizowana jest ona przez funkcję losującą 0 (wolna przestrzeń) i 1 (przeszkoda) z zadanym prawdopodobieństwem wystąpienia przeszkody.

W następnej części na podstawie wcześniej stworzonej macierz generowany jest graf połączeń. Wierzchołki grafu odpowiadają strukturze przeszkód, natomiast wagi przejść odpowiadają odległości między nimi. Algorytm pozwala na ruch w pionie i poziomie, uniemożliwiając wybranie ścieżki po skosie.

Ostatnim etapem jest funkcja wybierająca najmniejsza wagę przejścia między wierzchołkami, aż do przejścia do punktu docelowego.

2.2. Środowisko

Projekt został zaimplementowany w środowisku Python. Podczas pracy nad zadaniem wykorzystano takie moduły jak *heapq*, *random*, *multiprocessing*.

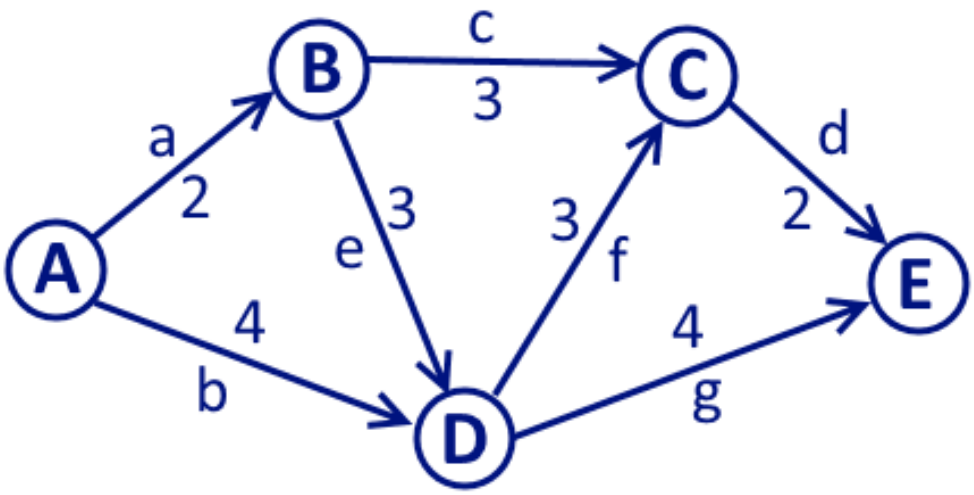
Moduł *heapq* jest zbiorem funkcjonalności pozwalających na operacje na strukturze danych (stosem). Pozwala m. in. na dodawanie/odejmowanie danych, pobieranie ich. W projekcie wykorzystana została przy wyszukiwaniu najkrótszej ścieżki w grafie. Przechowuje i aktualizuje kolejkę wierzchołków do rozważenia podczas przeszukiwania grafu.

Moduł *random* w Pythonie dostarcza funkcji związanych z generowaniem liczb pseudolosowych. W programie wykorzystany został przy generowaniu obszaru pracy robota.

Moduł *multiprocessing* w Pythonie umożliwia programistom wykonywanie równoległych i współbieżnych operacji. W projekcie wykorzystany został podczas generowania grafu połączeń. Dzięki jego zastosowaniu skrócono czas generowania grafu dla dużych plansz.

3. Opis algorytmu

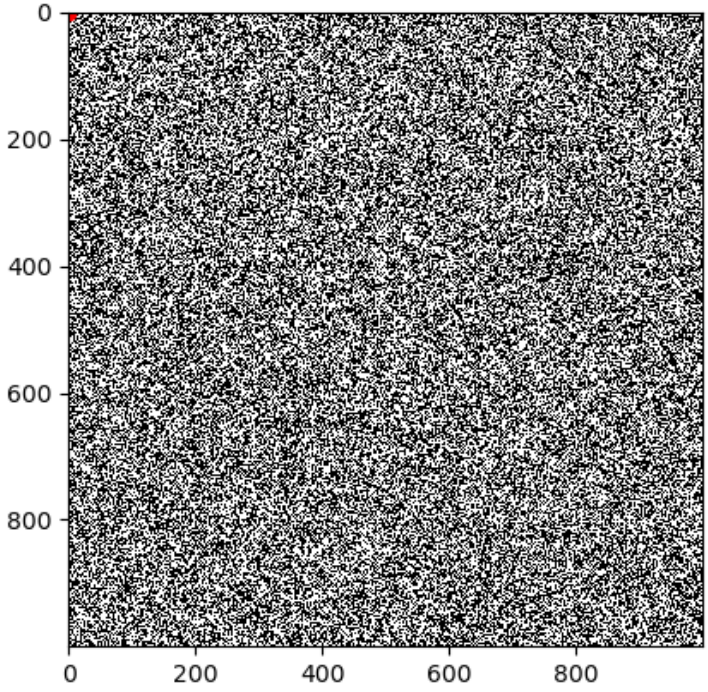
Algorytm Dijkstry to metoda służąca do znajdowania najkrótszych ścieżek między dwoma wierzchołkami w grafie z wagami na krawędziach. Jego celem jest ustalenie najmniejszych odległości od jednego wierzchołka, zwanego źródłem, do wszystkich pozostałych wierzchołków w grafie (rys. 3.1).



Rysunek 3.1. Przykładowy graf

4. Test programu

Do testu programu wykorzystano plansze o wymiarach 1000 x 1000 elementów (rys. 4.1). Program został przetestowany dla następujących prawdopodobieństwem wystąpienia przeszkody: 10%, 20% ,30%.

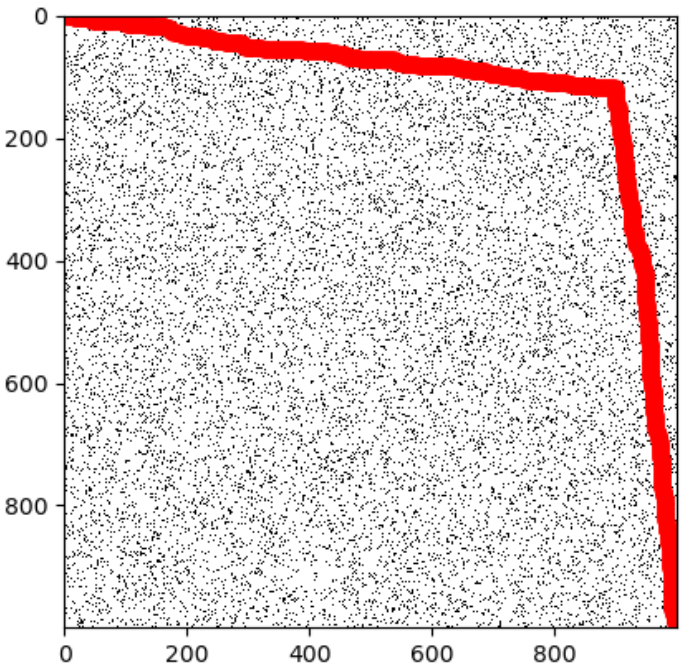


Rysunek 4.1. Przykładowy obszar roboczy

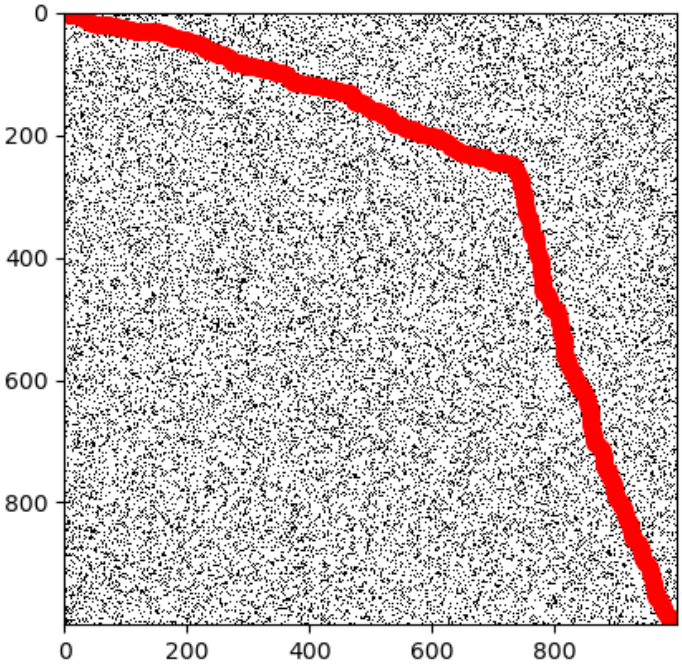
Dla każdego poziomu prawdopodobieństwa przeprowadzono 10 symulacji na podstawie których opracowano poniższe wyniki. Dodatkowo na rys. 4.2, rys. 4.3 i rys. 4.4 przedstawiono przykłady otrzymanych trajektorii.

Tabela 4.1. Wyniki symulacji

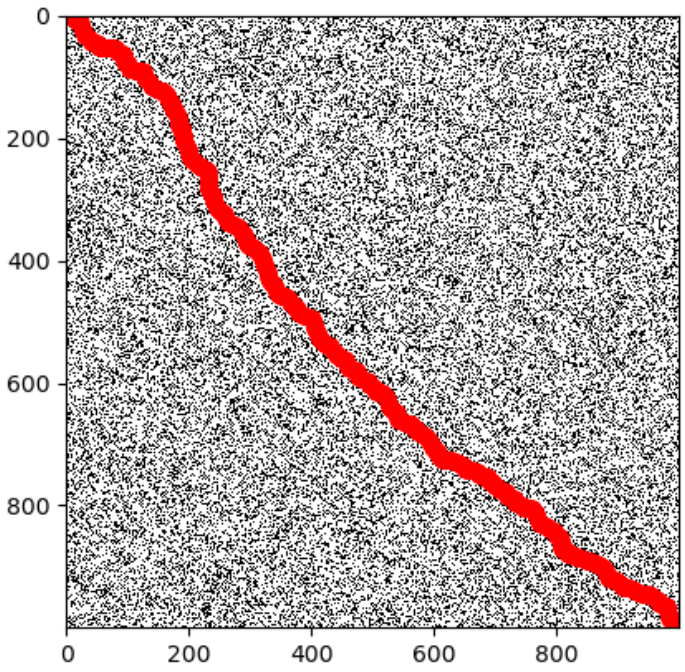
Prawdopodobieństwo [%]	10
Długość trasy [−]	1998
Czas wykonywania programu [s]	10,5
Prawdopodobieństwo [%]	20
Długość trasy [−]	1998
Czas wykonywania programu [s]	9,5
Prawdopodobieństwo [%]	30
Długość trasy [−]	2006
Czas wykonywania programu [s]	8,5



Rysunek 4.2. Przykładowa trajektoria dla 10%



Rysunek 4.3. Przykładowa trajektoria dla 20%



Rysunek 4.4. Przykładowa trajektoria dla 30%

5. Podsumowanie

Na podstawie wyników przedstawionych w poprzednim rozdziale można wywnioskować, że zwiększanie zagęszczenia przeszkód skraca obliczenia. Może to być spowodowane mniejszą liczbą połączeń węzłów grafu, przez co istnieje mniej możliwości przeprowadzenia trajektorii. Natomiast różnica w długościach tras jest niewielka. Podczas testowania programu dla największej gęstości przeszkód można było zauważyć częstsze nieznajdowanie trasy.

Dzięki wykonywanie obliczeń na maszynie wirtualnej oraz wykorzystaniu równoległych operacji można było przeprowadzić badania na większym, bardziej skomplikowanym obszarze roboczym.

Podczas przyszłej pracy nad projektem należałoby dodać funkcjonalność patrzenia w przód algorytmu. Dzięki takiemu działaniu można by skrócić trajektorię dla obszarów o mniejszej gęstości przeszkód oraz ograniczyć występowanie „ślepych uliczek".

Spis rysunków