Ztracený robot

Jan Kimr

ČVUT–FIT

kimrjan@fit.cvut.cz

3. května 2023

1 Úvod

V této práci jsem se zabýval problémem lokalizace robota. Robot se nachází na neznámém místě v bludišti s neznámou orientací. Robot zná mapu prostředí, ve kterém je umístěný, a má optické senzory s omezeným dosahem, které zaznamenávají jeho okolí. Cílem je nejrychleji (co do počtu pohybů a otočení) určit pozici (bod a orientaci) robota pomocí pohybu, otáčení a zaznamenávání okolního prostředí.

2 Vstupní data

Vstupem je jméno souboru s mapou bludiště. Dále je možné zadat pozici (bod a orientace) robota a dosah jeho senzorů. V případě, že pozice nebo orientace není zadaná nebo je ve stěně bludiště, pak je hodnota zvolena náhodně.

Pozice je reprezentována jako bod $[x, y]$, kde x reprezentuje řádek v mapě a y sloupec. Tuto reprezentaci jsem zvolil, jelikož je možné tyto souřadnice rovnou používat k přístupu do mapy. Orientace robota je buď číslo od 0 do 3, kde 0 je dolů a další směry jdou proti směru hodinových ručiček, nebo jednotkový vektor, který reprezentuje odpovídající pravoúhlý směr.

3 Algoritmus určení počáteční pozice

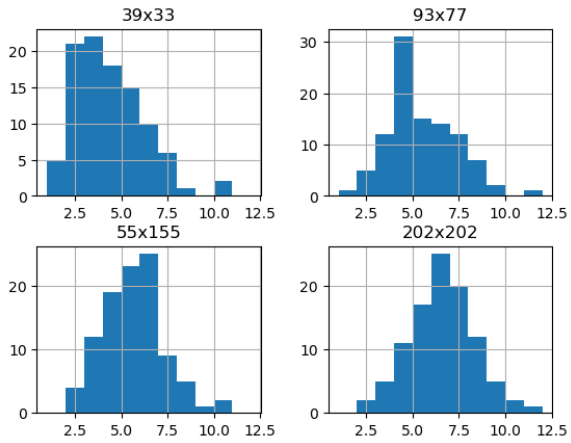
Robot zná mapu bludiště, ve kterém se nachází, a kromě toho má i svoji mapu, do které si po každém pohybu ukládá objevené okolí. Na začátku do této mapy přidá to, co právě vidí, a zjistí všechny možné pozice, na kterých může být. Následně najde nejbližší místo (co do počtu kroků), a to takové, že jeho navštívení sníží počet možných počátečních pozic alespoň o jedna. Poté, co robot na toto místo dojede, se proces opakuje a to až do té doby než zbývá pouze jedna možnost, nebo již není možné pozici zpřesnit (např. u symetrických bludišť). Seznam možných počátečních pozic je následně vrácen.

3.1 Určení bodu a cesty

Bod je určena tak, že z bodu a orientace ($[0, 0]$, 0) je puštěn Dijkstrův algoritmus, který postupně prohledává okolní body, na které může robot vstoupit. Tyto body jsou prohledávány v pořadí rostoucího počtu kroků, které by robot potřeboval k přesunu z pozice $([0, 0], 0)$ na pozici $([x_i, y_i], d_i)$. Pro tento bod a orientaci $([x_i, y_i], d_i)$ a každý počáteční bod $([x_{s_j}, y_{s_j}], d_{s_j})$ je určen bod $([x_{s_j}+x_i, y_{s_j}+y_i], d_{s_j}+d_i \mod 4)$, který reprezentuje možnou pozici a orientaci robota po provedení cesty odpovídající přesunu z $([0, 0], 0)$ od $([x_i, y_i], d_i)$. Pro takto získané pozice je otestováno, zda prostředí, které by robot viděl, je ve všech případech stejné. Pokud ano, pokračuje prohledávaní dále. Pokud ne, je dopočítána cesta do tohoto bodu. Robot se přesune na nalezenou pozici a postup se opakuje. Může se stát, že algoritmus doběhne (projde všechny možné pozice v bludišti), to znamená, že pozici již nelze zpřesnit.

4 Výsledky

Program jsem testoval na mapách z první úlohy předmětu ZUM [1] a náhodných pozicích. Jak je vidět na následujícím histogramu, ve většině případů stačilo okolo pěti až šesti kroků (posunů a otočetní), aby robot určil svoji pozici.



Obrázek 1: Histogram počtu kroků potřebných k určení pozice podle rozměrů mapy

5 Závěr

V rámci zpracování projektu jsem vymyslel a implementoval jeden způsob, jak robot může určit svoji pozici pomocí postupného eliminování možností. Tento postup se na mapách bludišť ukázal jako efektivní, jelikož k jednoznačnému určení pozice většinou stačilo okolo šesti nebo méně kroků. V rámci zpracování práce jsem se dost naučil, nicméně stále je zde prostor pro zlepšení, zejména zrychlení běhu programu na větších mapách.

Reference

[1] Klára Hájková. Datasety k úkolu: Systematické prohledávání stavového prostoru. online, 2019. [cit. 2023–05–03] `https://courses.fit.cvut.cz/BI-ZUM/labs/01/dataset.tar.gz` a `https://courses.fit.cvut.cz/BI-ZUM/labs/01/testovaci_data.zip`.