Dynamic Multi-Threaded Path Planning Based on Grasshopper Optimization Algorithm

Asmaa Shareef
Computer Science Dept.
College of Education for Pure Sciences
University of Basrah
Basrah, Iraq
asmaa.shareef@uobasrah.edu.iq

Salah Al-Darraji
Computer Science Dept.
College of Education for Pure Sciences
University of Basrah
Basrah, Iraq
aldarraji@uobasrah.edu.iq

***Abstract*— Over the past few years, computer hardware equipment has undergone a great deal of development, especially since multi-core processors were introduced. Compared to single-core, multi-core chips can provide higher performance and efficiency. In this paper, we exploit multi-core system to speed up path planning using the grasshopper optimization algorithm (GOA) as well as finding multiple paths supporting the mobile robot to navigate safely in a dynamic known environment. The multiple paths can be used if the robot encounters a dynamic obstacle to fabricate a new obstacle-free path out of the planned paths. Diverse experiments have been conducted using a random dynamic environment. The experiments have shown that the robot always finds the path to the target even with multiple dynamic obstacles that block the way of the robot.**

Keywords*— *path planning, GOA, multi-thread, dynamic obstacle, autonomous mobile robot.

I. INTRODUCTION

The ability to plan a path is one of the most important features of autonomous mobile robots. Planning a path can make moving the robot from the starting point to the target point much smoother and safer [1]. Research and studies have shown that a variety of methods and algorithms can be used to plan a path for a robot [2-4].

Global (offline) path planning is finding a path from a vast amount of data available, which is very effective when the environment is static and well-known to the robot. On the other hand, the local (online) path planning approach is used when an environment is unknown or dynamic, which is regarded as one of the biggest challenges in path planning. In this case, the robot uses local sensors while traveling from the start to the target point. As the environment changes, the robot may create a new path. Fig. 1 depicts the path planning classification criteria.

A robot's navigation is most accurate and efficient when path planning algorithms are used. Mac et al. [5] presented two path planning algorithms: classical and heuristic. The classical algorithms produced clear results but were extremely complex. Alternatively, heuristic algorithms are often constructed based on probability.

Recently, new studies have appeared in which optimization algorithms have been presented, which have contributed to great advancements in various fields, especially path planning, where the algorithms have solved most of the high complexity problems faced by classical algorithms. Khattar et al. [6] categorized optimization algorithms into heuristic and metaheuristic. Metaheuristic optimization algorithms are based on the natural phenomenon of living organisms. It has been proven that swarm intelligence algorithms are effective at path planning. Among the most famous swarm intelligence algorithms are Ant Colony Optimization (ACO) [7] and Particle Swarm Optimization (PSO) [8].

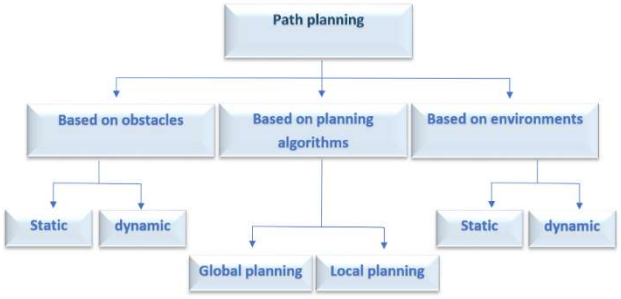

Fig. 1. Path planning algorithm classification criteria.

In 2017, Saremi et al. introduced the Grasshopper optimization algorithm (GOA) [9]. GOA is a meta-heuristic and bio-inspired algorithm based on population, which is used to plan a path based on the behavior of grasshoppers in obtaining food and achieving their goal.

Based on the resources of a multi-core system [10-12], this study builds a navigation system based on the GOA to decrease the convergence time in a dynamic environment. The contribution of this work can be summarized as follows:

- A multi-threaded path planning based on GOA is used to obtain multiple paths from start point to target point. This parallelization decreases the time and cost used to obtain these paths, which can be used later in navigation.
- During navigation, when a dynamic obstacle is encountered, the robot exploits the multiple paths to create an alternative obstacle-free path or backtrack to another path.