INTEGRATION OF A SYSTEM FOR SIMULTANEOUS UAV RELAY POSITIONING AND AREA COVERAGE IN C# AND UNITY

DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering

University of Genova

EMARO - European Master on Advanced Robotics

Maria Eduarda Andrada, Avgi Kollakidou

supervised by
Prof. Fulvio Mastrogiovanni
Alessio Capitanelli & Andrea Nistico

February 2018

Introduction

Unmanned Aerial Vehicles (UAV) have been in the center of attention of the scientific community in the past years. As technology advances and provides solutions to their limitations, new and more complex implementations are brought within our reach. Applications include: Surveying and Mapping of unknown areas, Agriculture, Traffic Surveillance and Monitoring of forest areas. this project focuses on the cooperation of multiple UAVs to achieve full area coverage while maintaining constant communication with a ground base station. This is implemented with the usage of a UAV relay, operating as a chain, both for path planning and communication purposes. The positioning of the UAVs is determined with a control algorithm, which in the future could be used in life threatening cases, such as search and rescue missions after earthquakes or other natural disasters.

For the project purposes, the area is considered known and mapped. More specifically, the area that is to be monitored by the UAV relay is represented as a grid map where grids are considered as nodes for the algorithm and they are either free or blocked (an obstacle exists). The UAVs can only occupy free nodes, but as they can communicate through obstacles, communication is not interrupted. An implementation of the Dijkstra Algorithm is used to calculate the shortest path between each pair of nodes and consequently from the ground-base to the target, which is set as the position of the last UAV in the relay. Finally the Dual Ascent algorithm is applied, to optimize the path and the general results.

Problem and Solution

The Shortest Path Problem

The shortest path problem is usually displayed with a directed graph and an arc or edge between each node represents the distance - commonly referred as weight - between them. The weight is calculated with a cost function depending on:

- distance between source and target
- time of travel
- cost of travel
- number of vehicles necessary

The Dijkstra Algorithm

The Dijkstra Algorithm was first designed by the computer scientist Edsger W. Dijkstra in 1956 and offers a solution to the shortest path problem. In its original form the algorithm calculates the shortest path between two nodes but few variations exist. In this project we implement a common variant where a single node is considered as the source and the algorithm calculates the shortest path to the target node.

The Dual Ascent Algorithm

The Dual Ascent is an iterative algorithm used to optimize our path planning. Dual Ascent in general is used for large scale optimization, where the constraints can be efficiently enumerated. The Algorithm takes into account the cost function, which consists of the number of vehicles used or the energy requirements for the flight and optimizes the solution converging to an optimal dual pair.

Unity

Unity is a cross-platform game engine which develops 2D or 3D simulation or games for computers, mobiles or consoles. It was developed by Unity Technologies. It can be scripted using C# or UnityScript and allows quick visual implementation of the script which is ideal for this project to facilitate the introduction of the algorithm to groups that are not in the Computer Science field.

Unity also provides a user friendly interface that aids to the easy and quick change of the project parameters. The visualization of the tools, drones or maps, can also be changed problem-free through the modularity of Unity.

Unity Navigation Guide

1. When the project is opened in Unity, the correct scene has to be selected in order to run the algorithm. In our case a single scene exists (map2). The Scenes, Scripts and all other project assets can be found in the "Project Window"

2. At the press of the play button Unity will enter into "Game Mode" and will display any changes made previously by the user and execute all relevant scripts.

3. Any changes made in the Interface while Unity is in Game Mode will not be conserved. The User has to press the play button again to return to Edit Mode and change any parameters needed.

4. All accessible variables are located in the Inspector Window. The game object "Map" has to be selected in the Hierarchy Window for the variables to be visible.

Implementation

The implementation was made in C# by creating a visual aid on Unity of the algorithm previously created by the supervisors. Modifications on the algorithm code were made to be able to apply it on Unity.

Unity Interface

The User can define the desired initial conditions in the Unity Interface. Variables that can be modified are:

1. The Map Size
2. Maximum number of additional vehicles to be used (a vehicle is always placed at source and target positions)
3. Position of Source - Home Base
4. Position of Target
5. Number of Obstacles (the position will be chosen arbitrarily)
6. Seed variable - provides the algorithm with a different arbitrary set of obstacles
7. Grid Outline (the outline of each grid can be changed if needed for better definition)

When done, visualization can begin with the press of the play button on the top of the window.

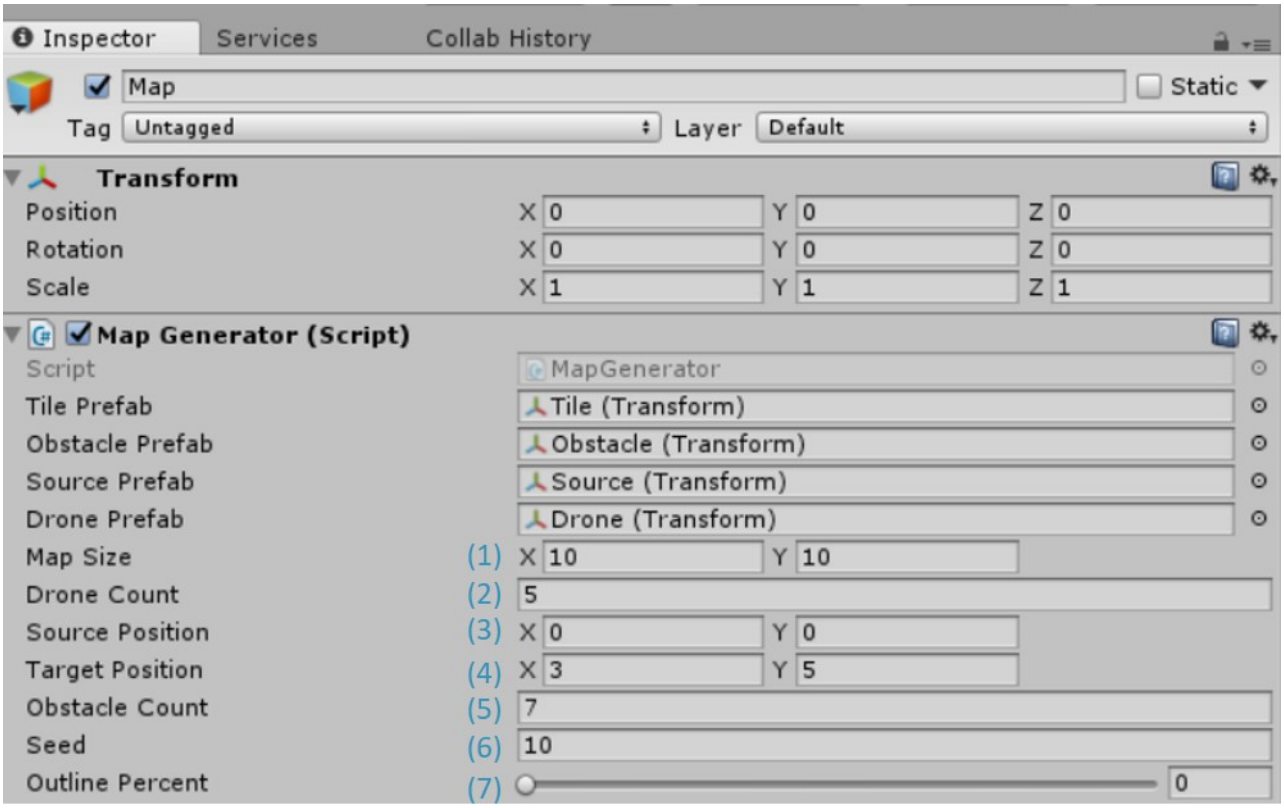

Figure 1: Inspector Tab in Unity Interface

Results

The control algorithm will be carried out with the parameters set by the user in the Unity interface. If they are satisfactory and the algorithm can deduct a path of continuous communication the Grid Map and positions of the Drones will be displayed (at the press of the play button). In case the algorithm fails to deliver a solution the Grid Map will be displayed with a failure notification.

Successful Implementation

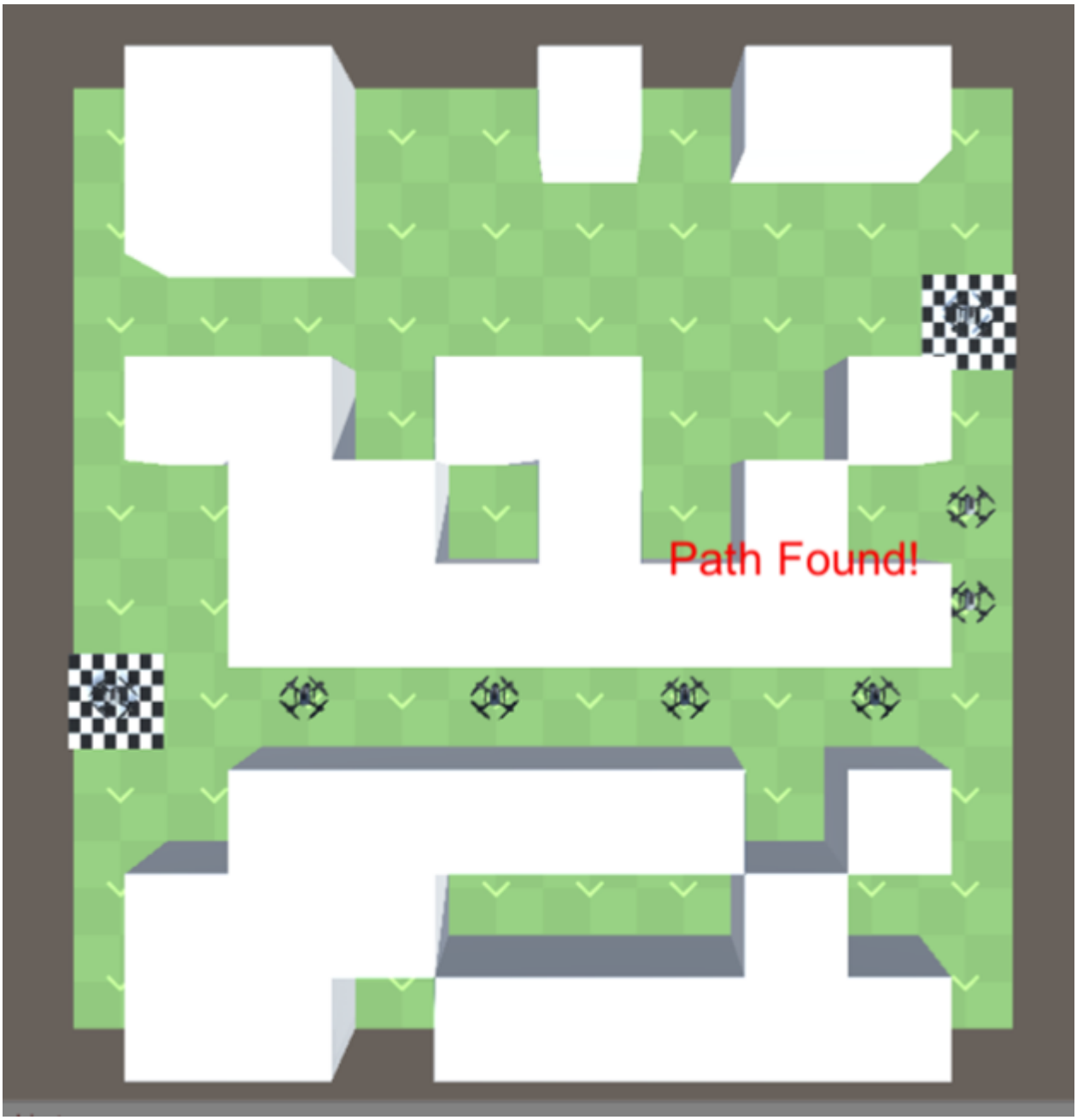

Figure 2: Display of successful control - Drone Path

Unsuccessful Control

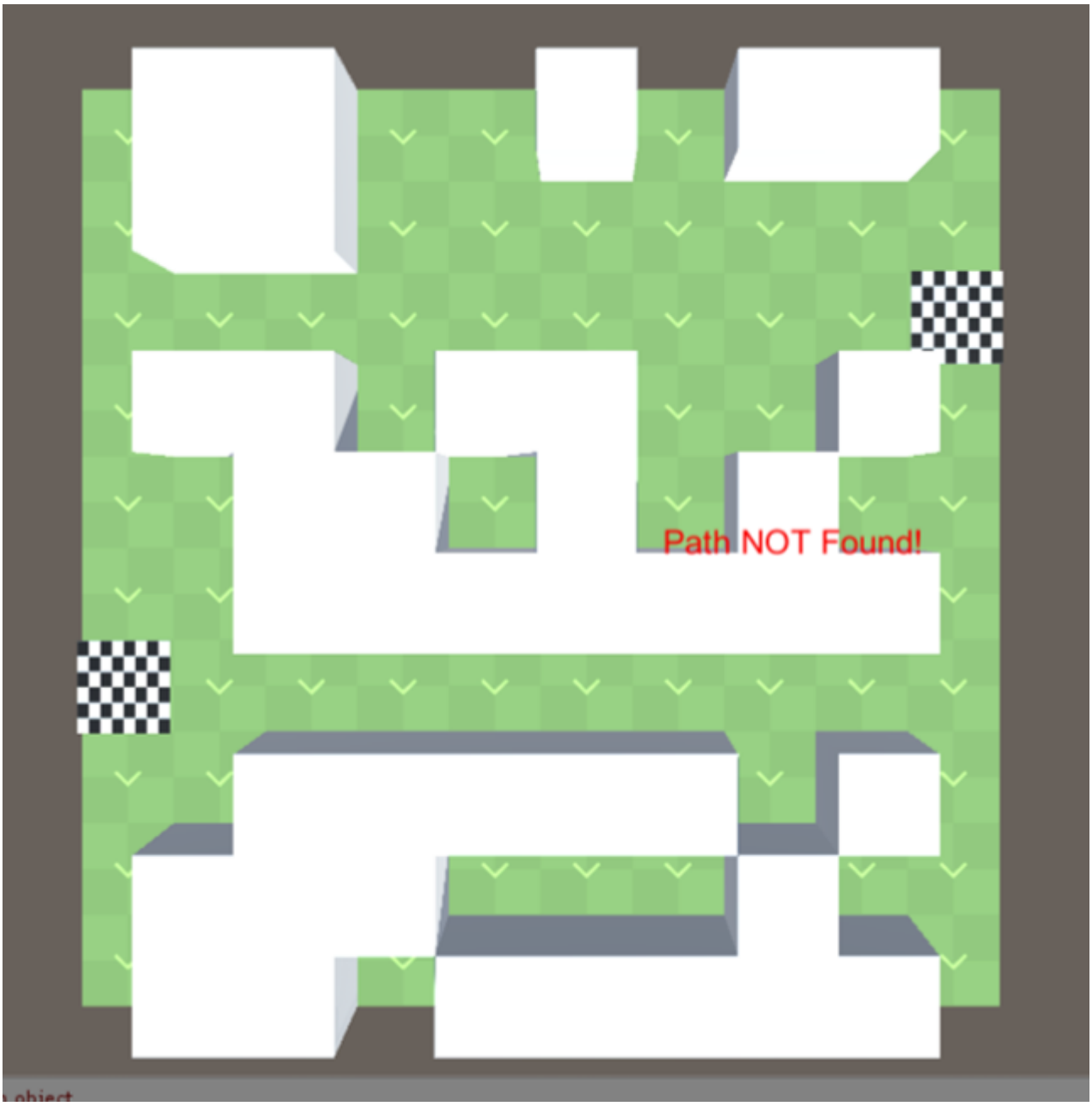

Figure 3: Display of Grid Map and notification of the algorithm's failure