An Intelligent Robotic System for Localization and Path Planning Using Depth First Search

Andrea Doucette[1] **and Wei Lu**[1,2]
[1]Department of Computer Science, Keene State College, USNH, Keene NH USA
[2]Department of Electrical and Computer Engineering, University of Victoria, Victoria BC Canada

Abstract - *LeJOS is an open source project created to develop a technological infrastructure and a tiny Java virtual machine to develop software into Lego Mindstorm Products using Java technology. LeJOS has been widely applied to programming Lego robotic system since it was created in 2006. In this paper we propose and develop a new depth first search algorithm and integrate it into the LeJOS system so that the intelligent robotic system is able to complete the localization and path planning automatically. Such a new depth first search algorithm is a significant improvement to the current path finding class provided by the LeJOS and experimental evaluations with the Mindstorm robotics system show that our approach can achieve the goal state intelligently in a short time.*

Keywords: LeJOS, Depth First Search, Java Virtual Machine

1 Introduction

Searching falls under Artificial Intelligence (AI). A major goal of AI is to give computers the ability to think, or in other words, mimic human behavior. The problem with this mimicry is that unfortunately, computers don’t function in the same way as the human brain: they require a series of well-reasoned out steps in order to find a solution. Therefore our goal is then to take a complicated task and convert it into simpler steps that the robotics system can handle. That conversion from something complex to something simple is what the search algorithm would do [1]. In this paper we propose and implement a new depth first search (DFS) on a tiny Java Virtual Machine (JVM), called LeJOS [2]. LeJOS is an open source project created to develop a technological infrastructure in which Java technology is applied for programming software for robots. Java is an Object Oriented programming language and one of the most important features implemented in LeJOS is the *LeJOS navigation API* that can be used to achieve the goal in which a convenient set of classes and methods provided to control the robot. The classes that control vehicles deal with several levels of abstraction. At bottom, there are the motors that turn the wheels, controlled by the *NXTRegulatedMotor* class. The *DifferentialPilot* class uses the motors to control elementary moves: rotate in place, travel in a straight line, or travel in an arc. At the next level, the *NavPathController* uses a *DifferentialPilot* to move the robot through a complicated path in a plane. To perform navigation, the path controller needs the robot location and the direction it is heading. It uses an *OdometeryPoseProvider* to keep this information up to date.

The contributions of the paper mainly include (1) a new depth first search (DFS) algorithm that can be applied to build arbitrary tree structures generically and (2) applying and integrating the proposed DFS algorithm in the LeJOS based robotics system for localization and path planning, enhancing the existing pathfinding approaches within LeJOS system. The rest of the paper is structured as follows. Section 2 introduces the depth first search algorithm. Section 3 presents our proposed leJOS based DFS algorithm. Section 4 outlines the experimental evaluation settings and results for our system. Finally Section 5 makes concluding remarks and outlines our future work.

2 Overview of depth first search

Let's first learn how we humans would solve a search problem. First, we need a representation of how our search problem will exist. The following Figure 1 is an example of our search tree. It is a series of interconnected nodes that we will be searching through:

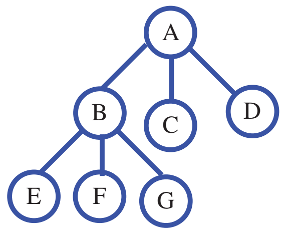

Figure 1. Tree structure of a path.

Depth first search works by taking a node, checking its neighbors, expanding the first node it finds among the neighbors, checking if that expanded node is our destination, and if not, continue exploring more nodes. For example if we want to find a path from A to E, we can use two lists to keep track of what we are doing - an *open list* and a *closed List*. An Open list keeps track of what you need to do, and the Closed List keeps track of what you have already done. At the beginning, we only have our starting point, node A. We haven't done anything to it yet, so let's add it to our *open list*. Then we have *open list* including <A> and *closed list* including <empty>. Now, let's explore the neighbors of our A node. Node A's neighbors are the B, C and D nodes. Because we are now done with our A node, we can remove it from our *open list* and add it to our *closed list*. Then our current *open list* include <B,C,D> and *closed list* contain <A>. Now our *open list* contains three items. For depth first search, you always explore the first item from our *open list*. The first item in our *open list* is the B node. B is not our destination, so let's explore its neighbors. Because we have now expanded B, we are going to remove it from the *open list* and add it to the *closed list*. Our new nodes are E, F and G, and we add these nodes to the beginning of our open list. Then we have *open list* including <A,B> and *closed list* including <E,F,G,C,D>. We now expand the E node. Since it is our intended destination, we stop. Therefore we receive the route A->B->E that is interpreted from the *closed list* by using the regular depth first search algorithm.

3 LeJOS based DFS algorithm

In our LeJOS based DFS algorithm each node on the path is a class node called WPNode defined as:

```
public WPNode(String newname, WayPoint newwp) {
   nodename = newname;
   nodewp = newwp;
   seen = false;
   parent = this;
   connections = new ArrayList<WPNode>();
}
```

The pseudocode for the LeJOS based DFS algorithm is described in the following:

(1). Constructing the generic tree for search space, such as:

```
A = new WPNode("A", new WayPoint(0, 0));
B = new WPNode("B", new WayPoint(-5, 5));
C = new WPNode("C", new WayPoint(5, 5));
A.addLeaf(B);
A.addLeaf(C);
```

(2). Declare a stack to save the route path, such as:

```
Stack<WPNode> DFSpath = new Stack<WPNode>();
```

(3). Set the current node to root node, say *A*. While the destination node is not found, loop the following:

a. if current node has children, set first node unseen node to current node then return;

b. if current node has no unseen children, set its parent to current node then return;

(4). Once the destination node is found, push the destination node to the stack and then push each parent node to the stack;

(5). Generate pilot for two-motor movement and then set pilot to use appropriate dimensions and motors

(6). Pop waypoint of each path node and apply goto method to direct robots moving to the next node.

The completed source code package can be found at [3] in more details.

4 Experimental evaluation

As illustrated in the Figure 1, we evaluate our Mindstorms robotic system to find the path from both *node A* to *node G* and *node A* to *node F*. Lego Mindstorms NXT is an educational product designed to build easy robots with an intelligent, computer-controlled NXT brick that lets a robot come alive and perform different operations [4]. In our experiment, the coordinate at A is (0,0), the coordinate at B is (-5,5), the coordinate at C is (0,5), the coordinate at D is (5,5), the coordinate at E is (-10,10), the coordinate at F is (-5,20) and the coordinate at G is (0,20).

The path from *node A* to *Node G* is A → B → G and path from *node A* to *Node F* is A → B → F, the Mindstorms robotics system with LeJOS DFS algorithm traverses each node until it reaches the destination G and F, respectively, showing the algorithm works successfully in the LeJOS JVM.

5 Conclusions

In this paper we propose and implement a LeJOS based DFS algorithm running under the LeJOS JVM. Experimental evaluation shows that our algorithm works successfully to achieve its goal within a short time. In the near future we will implement the Breadth First Search (BFS) and Heuristic based Hill Climbing algorithms to improve the current LeJOS path finding class. To the best of our knowledge the proposed work is the first attempt to improve the path finding class under the LeJOS JVM by integrating more generic searching algorithms in the LeJOS API functions.

6 References

[1] Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

[2] LeJOS, http://www.lejos.org/ retrieved in Mar. 28 2015.

[3] LeJOS DFS, http://sl.keene.edu/lejosDFS.zip

[4] Lego Mindstorms NXT 2.0, http://www.lego.com