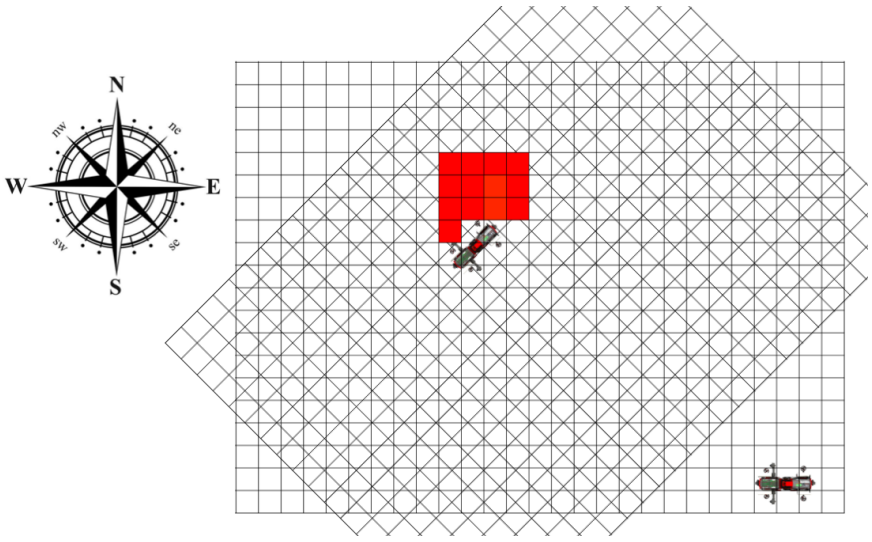
CU AUV

Cornell University
Autonomous Underwater Vehicle

Fall 2017

Positional Search



Semester Review

Noel Picinich (nmp53)

December 3, 2017

Contents

1 Abstract

The Positional Search project is an approach to locating a mission element that has not been seen yet. This implementation will standardize how we search for specific objects in each mission and store basic information about if and where an object was seen, thus, preventing the sub from restarting a hard coded search if the object is lost for a frame. This project will exist in the mission framework, with a specific search approach for each mission element in the competition.

This framework will allow software members, when writing missions, to create a specific search from the positional search framework that will locate the mission element for the respective mission and store a grid-like representation of space holding information about whether the mission element was seen at any given location. The dimensions of the cubes of space that compose this grid are defined by the user, allowing for smaller or larger search increments. In the long-term, this grid will also give priorities (as probabilities) to each cube in space, and the path traversed will be dictated by searching cubes with a greater probability of having a view of the mission element. This paper will focus on the initial implementation of this project which aims to be a functional search for the pipe mission element.

2 Goals

The primary goal of the positional search project is to implement a framework for standardizing how the sub searches for mission elements at the beginning of a mission. This framework, in full execution, will execute a more efficient search by following a search path that prioritizes the region of space with the highest probability of containing the desired mission element. Due to the specificity of each element search and the extensive testing required, this project will stretch into next semester as well.

2.1 Goals for the Fall 2017 semester

- Define and implement the structure of the Positional Search framework
- Produce a functional Downward Search class for the Pipes mission
- Test and iterate the Pipe_Search class in the simulator and in the pool

2.2 Milestones for the Fall 2017 semester

- Research searching algorithms
- Define what information the sub has about its velocity, heading, etc. and find a way to convert these values into a meaningful position
- Determine how to record and store information about an object when it is in sight
- Implement an efficient algorithm for the sub to search for a task, given the information obtained since the start of the mission.
- Test in the simulator and pool and make improvements

3 Previous Implementations

Since our software stack has not previously had a positional search framework, searching for mission elements was done by a series of hard-coded movements and conditionals. The major downfall in this method is the sub's inability to recover sight of a mission element that was previously seen when sight of the object is lost for a frame. The positional search framework aims to ameliorate this problem by storing information about where in space a specific mission element was searched for and whether or not it was seen at such locations.

4 Design

A high-level overview of the positional search framework is, given a mission element keyword and an anticipated distance away, a search class for the specific element will execute a specialized search for the element, while storing information about whether the object was seen in each division of space searched.

4.1 Hash Table Approach

This information is stored in a hash table, a data structure chosen for its accessibility speed. The keys for the hash table are Cube objects that represent divisions of space with dimensions defined by the unit attribute of the Search Grid class. This unit variable is set to a default value of 1 meter but can be changed by the user. Figure 1 shows the basic hierarchy of the framework.

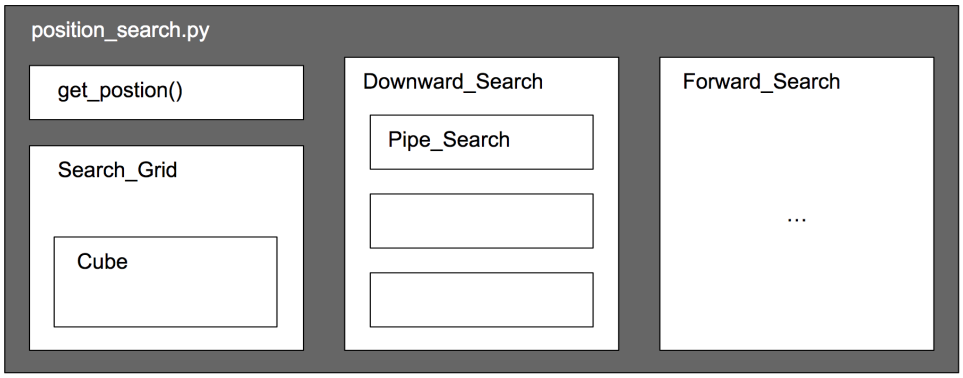


Figure 1: Depiction of the positional search framework design structure.

The hashcode that identifies the cube is the north, east, and depth values that identify the cube's position in space. These values are not the raw shm values, rather, they are a transformation relative to the sub's heading when the search was initialized, converted to the unit attribute, and truncated.

4.2 Search Classes

The Downward Search and Forward Search are the parent classes that users will call to create a search object. These classes create a specific search class depending on the key word the user passes in. In the case that the "pipes" keyword is specified, a Pipe Search object is created, which executes a search

for the pipe mission element. Since the search classes subclass Task, another framework often used in our missions, the Pipe Search class and other search classes are structured to satisfy the Task framework.

5 Implementation

The following sections describe the classes that form the positional search framework and how they fit together.

5.1 Search Grid

The search grid class stores a representation, in the form of a hash table, of the three dimensional space the sub has searched by storing weather the designated object has been seen in each cube of space with dimensions corresponding to the value of the unit attribute, the key corresponding to the box object that represents such cube of space, and the value corresponding to whether or not the desired object was seen when the sub was in such box. The Search Grid contains functions for determining whether a specific cube has been visited, setting the value of a cube key in the hash table, and calculating the cube that corresponds to a given position. Ultimately, the search grid class stores the hash table and the means to update and access it.

5.2 Cube

The cube class is a class nested in the search grid class to define the divisions of space that act as keys in the search grid hash table. As stated in the design section, the cube class redefines raw position values by transforming and converting them into integers that identify the cube of space relative to the subs heading and position when the search began. This class also contains functions to determine whether a given position lies within its boundaries and to return the cube object a specified distance away.

5.3 Downward Search

Downward search is the class called by the user on incorporation of the positional search framework. This class handles which search class to call given it's corresponding key word (see Figure 2).

```
def on_first_run(self, *args, **kwargs):
    mx = MoveX(self.distance)

    if self.task == "pipe":
        search_task = PipeSearch()
    elif self.task == "bins":
        search_task = BinSearch()
    elif self.task == "recovery":
        search_task = RecoverySearch()
    else:
        print("Incorrect input, choose from pipe, bins, and recovery")

    full_task = Sequential(mx, search_task)

    self.use_task(full_task)
```

Figure 2: Depiction of how Downward Search handles keywords.

Additionally, since the first step in every downward search is to travel the distance the mission element is anticipated to be from the sub, the downward search class executes this movement. On creation of a downward search object, a search grid object is also created, allowing the user to reference this attribute for the information stored in its hash table.

5.4 Pipe Search

Pipe search executes the motions to search for the pipe mission element. The search approach is to spiral out from the location with greatest probability of having the mission element. The sub does this by retrieving the cube one unit ahead and, if this cube has not been visited, moves to this cube and turns right, otherwise, turns left. After moving to a new cube, the sub checks for the pipe and if the pipe is found, terminates the search. In order to check if the pipe is in sight, the pipe search class utilizes a helper function, is_pipe_found, which returns true if the pipe results heuristic score is above zero.

6 Results

My work this semester has produced the basic functionality for a framework that will greatly improve the mission writing process and the efficiency of how the sub searches for mission elements. While still in its early stages, the positional search framework outlines a clear vision for how the remaining mission element search classes will be implemented, and the approach to storing information and representing space in a meaningful way has been implemented, tested, and iterated. As described in the future improvements section, there are vital troubleshooting and pool testing to be done before the framework can be reliably incorporated into our missions.

COMPLETE: The Search_Grid hash set for representing space as (user-defined) unit-dimensional cubes
COMPLETE: The Cube class and its transformation functions (Cubes act as keys in the Search_Grid hash set)
COMPLETE: A preliminary implementation of the Pipe_Search class
IN_PROGRESS: Simulator testing of Pipe_Search and an improved Pipe_Search iteration
INCOMPLETE: Pool testing of Pipe_Search and an improved Pipe_Search iteration

Figure 3: Progress report of semester goals.

7 Future Improvements

As previously stated, this semester focused on a high level implementation of the positional search framework, with a focus on the search class of the pipe mission element. In coming semesters, I plan to implement searches for the remaining mission elements, as well as incorporating probabilities and searching algorithms customized for a given mission element. Furthermore, the current iteration is yet to be tested in the pool, which must be a heavy focus for the framework to be a reliable resource. The lack of pool testing leaves major improvements to be made in the aspect of trouble shooting, that is, handling the events where an object is not found or there is a false positive.