

# PATH-FINDING FOR THREE DIMENSIONAL SPACES

P. J. Campbell

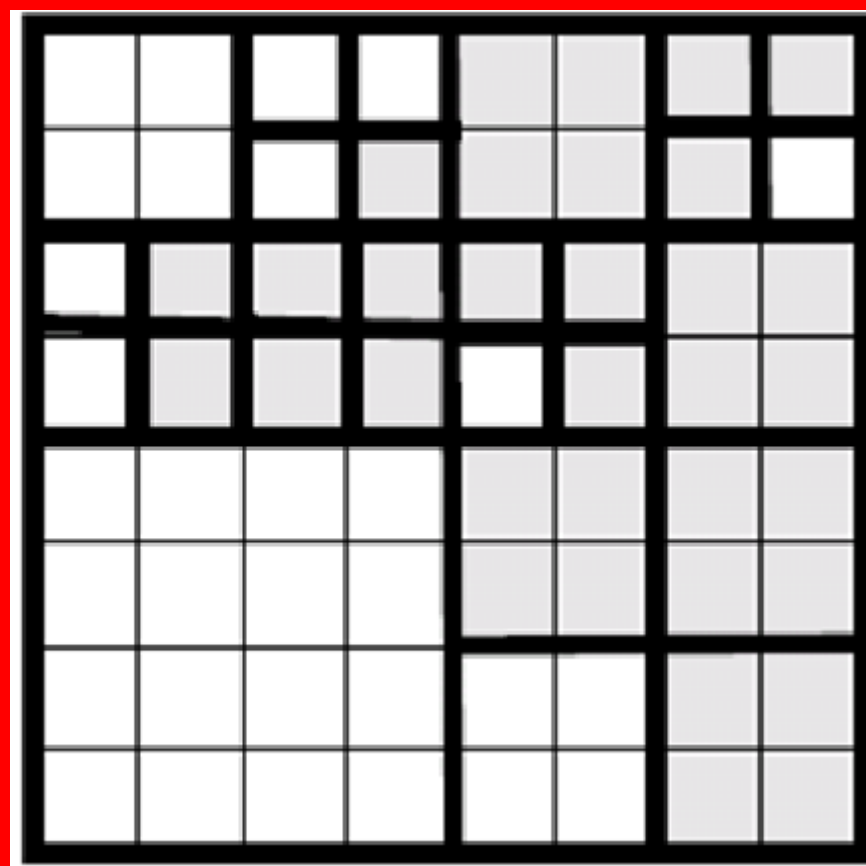
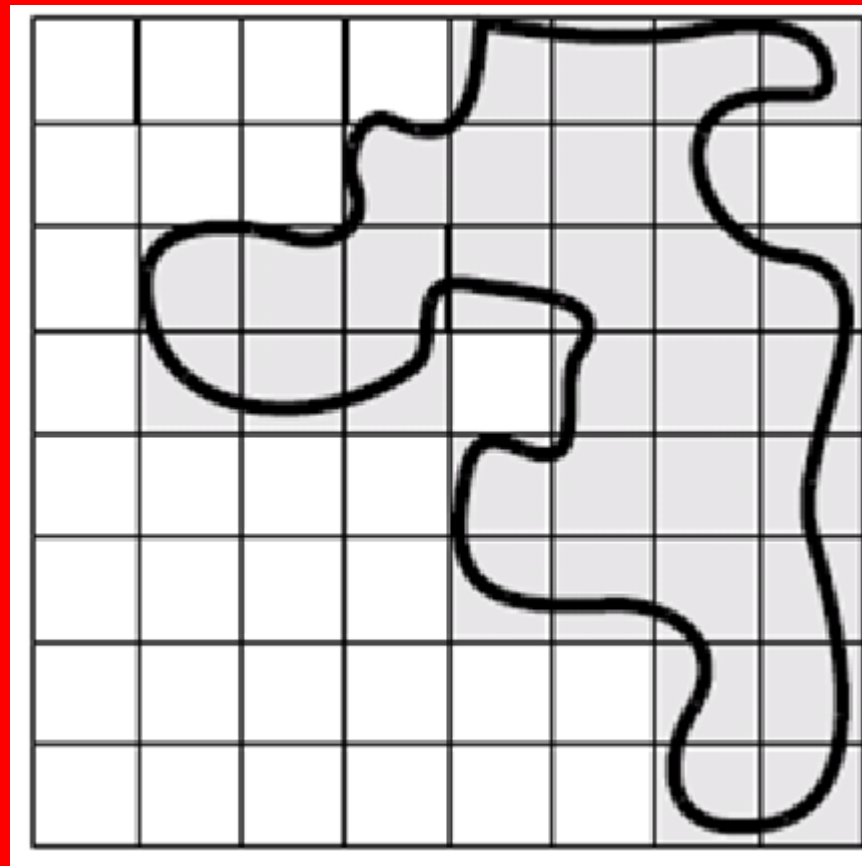
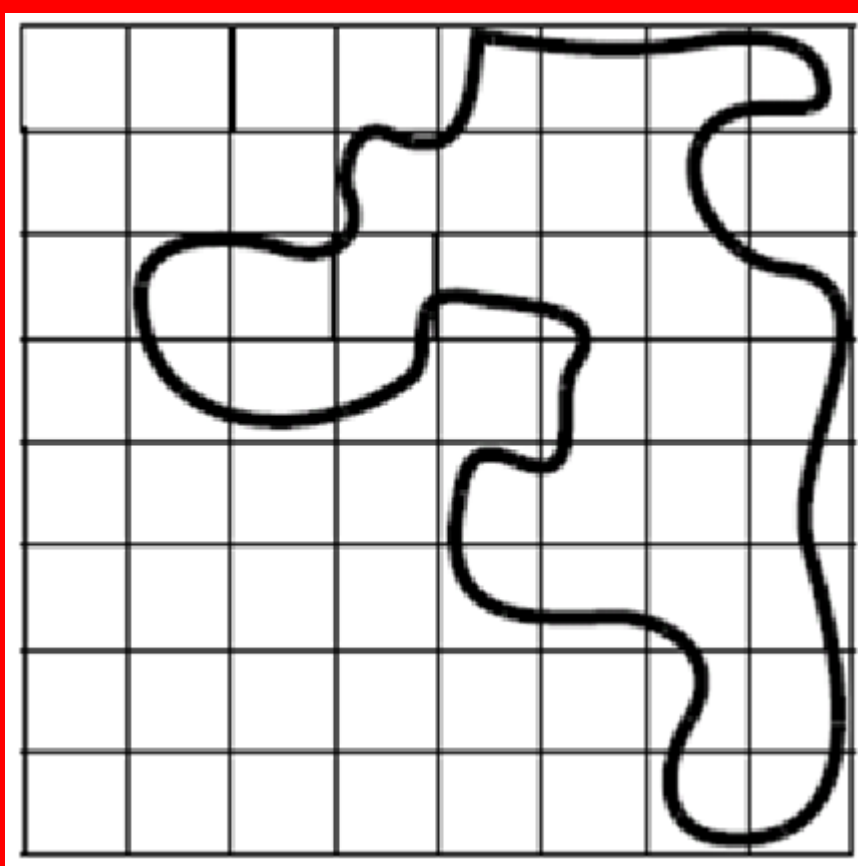
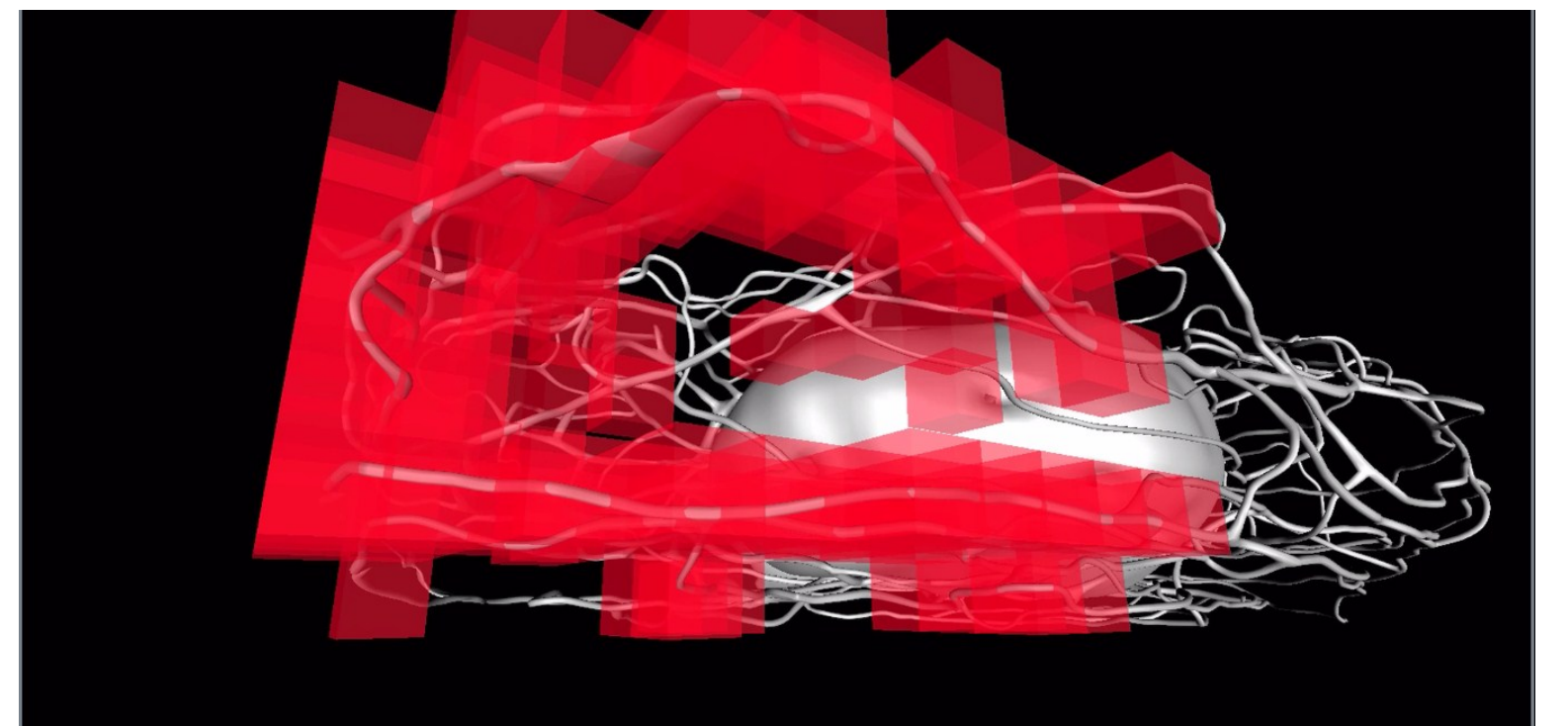
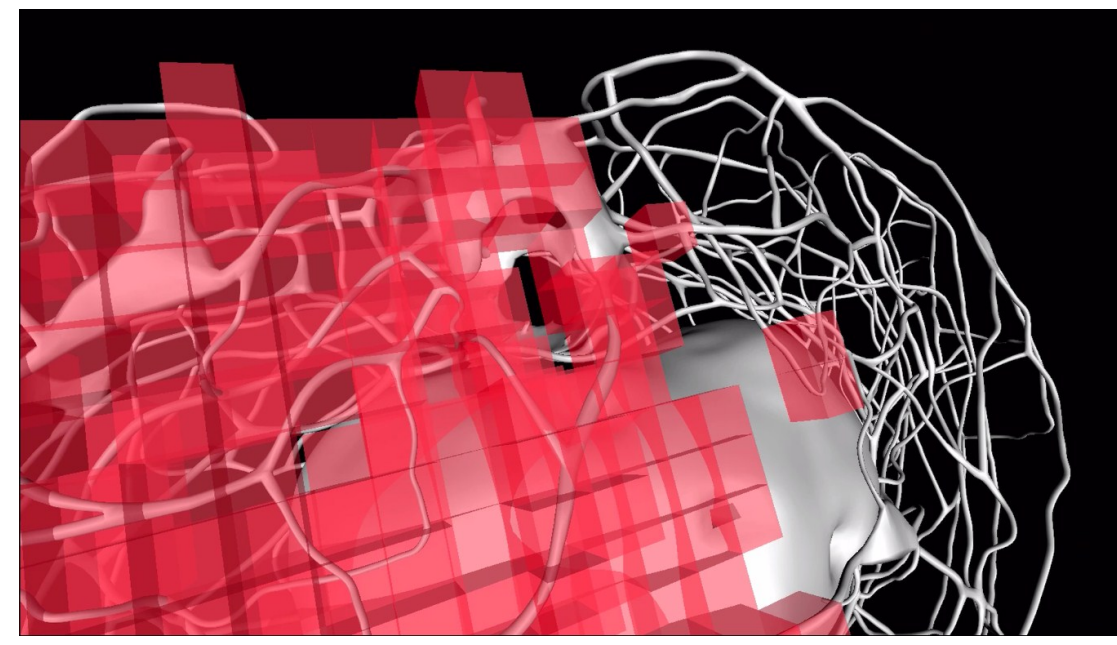
Major Professors:

Drs. Eve Wurtele & Julie Dickerson

## OCTREES

An octree is a graph in which the nodes of the graph have exactly eight children, representing the eight octants or regions of a 3D space. The root node of an octree contains the entire scene or domain.

Each node in an octree stores a point representing the center of the node, a pointer to its parent node, and pointers to its child nodes, if any. The parent of the root node is null. In addition, each node stores a list of its neighbors and a Boolean value expressing if there are any scene objects contained in it. A node is split into eight octants if it contains objects in the scene. The octree is constructed recursively until nodes contain a void, they reach their minimum volume allowed, or the maximum depth is reached.

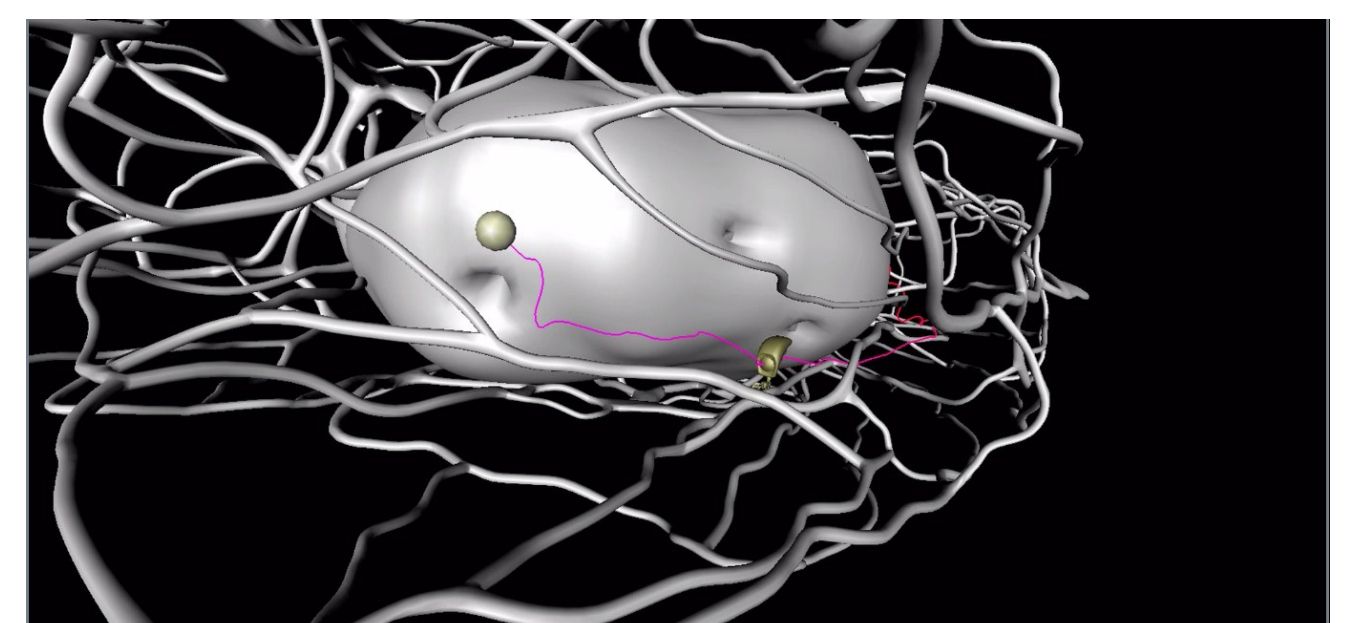
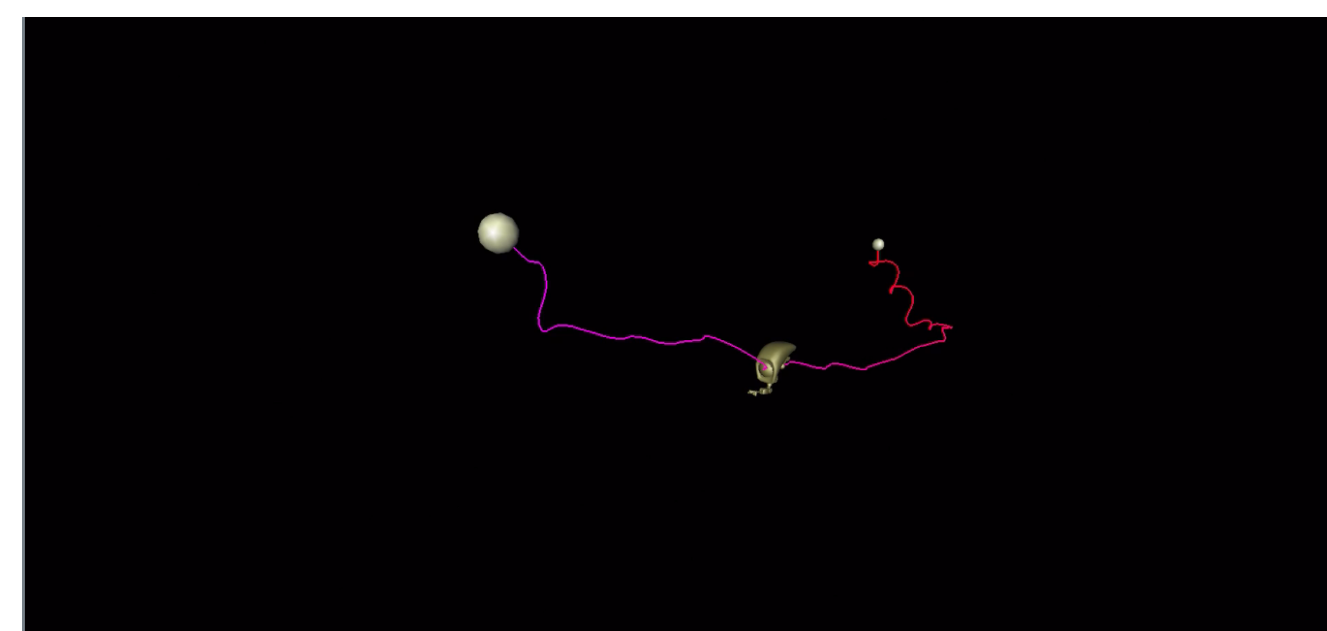


## EFFICIENCY

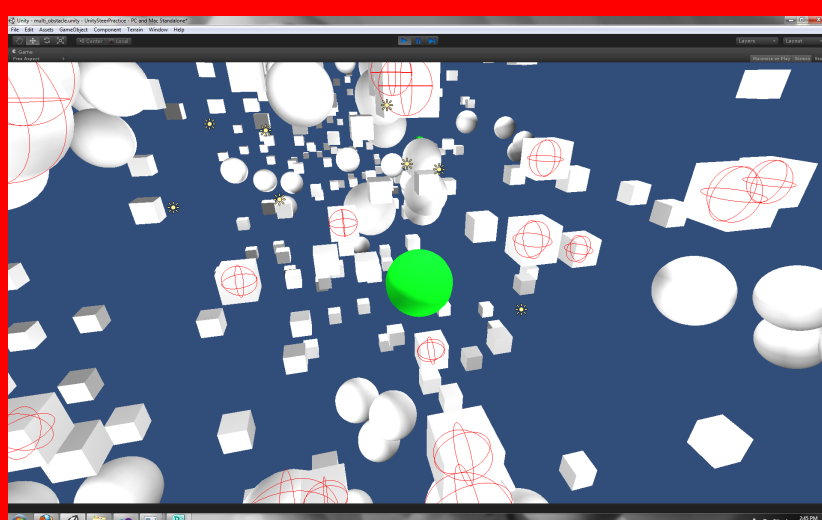
In this two-dimensional scenario, it shows that octrees are often more efficient than using a uniform grid. The scene contains an obstacle with an irregular shape. The first image shows how a uniform grid partitions the space and the second image displays the nodes in the grid that are not occupied by the obstacle, which are not highlighted with gray. The void in the lower left corner is represented with 16 grid nodes. But in the third image using the octree, this void is represented with only one node. This means our search algorithm has 15 less nodes to explore.

## A\* SEARCH

In combination with the octree data structure, we use the A\* search algorithm to traverse the graph. It searches the leaf nodes of the graph to find a path between an agent and a goal, if one exists. This example shows how the octree partitions a plant cell, the path generated by A\* and how the Nanobot character moves around the cell structure.



## FUTURE WORK



A set back for using the octree data structure is it takes 60 seconds to complete construction. Future direction with this work is to optimize computation time. Another direction is to combine this work with steering behaviors. As shown on the left, a steering behavior calculates the next collision with the nearest obstacle. Then uses this information to "steer away". Using this work, we can represent irregular obstacles more efficiently and generate better steering.

